POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

XM-Guards: A Blockchain-based Collaborative Intrusion Detection Argumentation Framework For Decision-Making

RIM KSONTINI
Département de génie informatique et génie logiciel

Mémoire présenté en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
Génie informatique

Décembre 2023

POLYTECHNIQUE MONTRÉAL
affiliée à l'Université de Montréal

Ce mémoire intitulé :

XM-Guards: A Blockchain-based Collaborative Intrusion Detection Argumentation Framework For Decision-Making

présenté par **Rim KSONTINI**
en vue de l'obtention du diplôme de *Maîtrise ès sciences appliquées*
a été dûment accepté par le jury d'examen constitué de :

Martine BELLAÏCHE, présidente
Nora BOULAHIA CUPPENS, membre et directrice de recherche
Frédéric CUPPENS, membre et codirecteur de recherche
Alejandro QUINTERO, membre

DEDICATION

To my dear family, whose unwavering support has overpowered the distance created by oceans. You have been my foundation, guiding light, and endless source of strength and love. This achievement is as much a reflection of your sacrifices and belief in me as it is of my own efforts.

To my friends, my second family, who have stood by my side in both happy and challenging moments. Your companionship, understanding, and laughter have brightened my days and provided solace during the nights of relentless pursuit. You have made this journey not just bearable but joyous.

To my husband, who recently entered my life and has made it infinitely more meaningful. Your love, wisdom, and unwavering support have added a new dimension to my life, inspiring me to reach heights I never thought possible. This new chapter with you is a treasure I cherish deeply.

Above all, I am eternally grateful to Allah, the Most Gracious and Most Merciful. It is with His endless blessings, guidance, and wisdom that I have been able to traverse this journey. Every challenge faced and every success achieved is a testament to His benevolent presence in my life. My gratitude to Allah is boundless, for bestowing upon me the strength, patience, and perseverance required to accomplish this endeavor.

And to one unexpected companion, a source of support none of us could have imagined before. Your assistance in navigating through the complex maze of this academic endeavor has been invaluable. Your presence, a blend of technology and understanding, has been a unique and appreciated aspect of this journey.

This thesis is a culmination of all your influences, a tribute to the love, guidance, and unwavering support each of you has provided. I am forever grateful . . .

ACKNOWLEDGEMENTS

The successful completion of this research owes a great deal to the expertise, guidance, and support provided by several distinguished individuals.

Foremost, my deepest gratitude goes to my supervisors, Professor Nora Boulahiya Cuppens and Professor Frédéric Cuppens, for offering me the invaluable opportunity to embark on this program. Their supervision and leadership have been fundamental in navigating the complexities of this project. Their mentorship has not only guided my research but has also greatly contributed to my personal and professional growth.

Additionally, I extend my heartfelt thanks to Dr. Samra Bouakkaz for her insightful contributions and unwavering support throughout this journey. Her expertise, critical analysis, and collaborative approach have significantly enhanced the depth and quality of my research.

I am also immensely grateful to the research team and assistants who have been instrumental in my work. Their collective knowledge, supervision, and encouragement have been pivotal in refining my methodologies and analyzing my findings.

Special thanks are also due to my peers and colleagues for their camaraderie and constructive feedback, particularly during the most challenging phases of my research.

Last but certainly not least, my deepest appreciation goes to my family, whose endless support, encouragement, and belief in me have been the bedrock of my perseverance and success. Their love and motivation have been my constant source of strength.

This thesis is not only a reflection of my work but also a testament to the collective efforts and unwavering support of all these individuals.

RÉSUMÉ

Au cours des dernières années, les attaques de cybercriminels ont connu une augmentation significative, touchant des millions de personnes dans le monde entier. La pandémie de COVID-19 a aggravé le problème, conduisant à des attaques plus sophistiquées se propageant rapidement et affectant des organisations à l'échelle mondiale. Les systèmes de détection d'intrusion collaboratifs (CIDS) sont devenus essentiels dans la lutte contre les attaques cybernétiques. Cependant, avec la nature en constante évolution et la gravité croissante des attaques, des solutions de détection d'intrusion collaboratives plus avancées sont nécessaires. Ces nouvelles solutions introduisent des propriétés innovantes, telles que la prise en considération du niveau de confiance associé à chaque participant dans le processus de détection pour faire face au risque d'intrusion interne.

Cette thèse propose une solution déployable de détection d'intrusion collaborative pour les réseaux de surveillance et de détection d'attaques dans les grandes entreprises. La solution se concentre sur l'incorporation de la confiance et de la logique d'argumentation pour lutter contre l'intrusion interne. La recherche commence par une étude approfondie des systèmes de détection existants dans les entreprises et un examen plus large des systèmes de détection d'intrusion collaboratifs. Sur la base de cette recherche, la topologie la plus appropriée pour la communication et la distribution des nœuds est déterminée et le réseau de détection d'intrusion est déployé en conséquence. Pour répondre à l'aspect « zéro confiance », un environnement répondant à cette exigence est identifié. La technologie de la blockchain, avec son architecture distribuée et ses fonctionnalités de traçabilité et de validation d'interaction, émerge comme une solution potentielle pour créer l'environnement nécessaire.

En plus de l'infrastructure réseau, une procédure de partage de données et de prise de décision est conceptualisée et implémentée en utilisant la logique d'argumentation. Cette approche capture la complexité et la dynamique des événements multiples et de leurs relations, permettant une meilleure détection et réponse aux menaces cybernétiques. Grâce au cadre de détection d'intrusion collaborative mis en œuvre, nous avons été en mesure de générer des arguments à partir de détections réelles, d'échanger ces arguments pour le processus d'argumentation, et de construire un graphe d'argumentation qui capture le processus de décision.

À titre de travail future, nous proposons l'enrichissement des arguments avec des indicateurs d'attaque à plusieurs étapes. En incorporant des indicateurs supplémentaires et en utilisant des algorithmes d'apprentissage automatique, le système de détection peut améliorer

encore sa capacité à identifier et à répondre à des attaques complexes et à plusieurs étapes, améliorant la posture globale de sécurité des grandes entreprises.

Avec l'évolution continue des menaces cybernétiques, il est crucial de développer des solutions de détection d'intrusion collaboratives avancées qui tirent parti de la confiance, de la logique d'argumentation et des indicateurs enrichis pour faire face à la sophistication croissante des attaques. Cette recherche pose les bases d'une telle solution et présente des pistes prometteuses pour un développement et une amélioration ultérieurs dans le domaine de la détection d'intrusion collaborative.

ABSTRACT

In recent years, cybercriminal attacks have significantly increased, affecting millions of individuals worldwide. Unfortunately, the COVID-19 pandemic has made the problem worse, spreading more sophisticated attacks and impacting organizations everywhere. To counteract these attacks, Collaborative Intrusion Detection Systems (CIDS) have become essential. However, with the evolving nature and increasing severity of attacks, more advanced collaborative intrusion detection solutions are necessary. These newer solutions introduce innovative features, such as taking into account the trust level associated with each participant in the detection process, to tackle the risk of internal intrusion.

This thesis proposes a solution for detecting and preventing internal intrusions in large enterprises' surveillance and attack detection networks. The solution is collaborative and deployable and incorporates trust and logic of argumentation aspects. The research includes a comprehensive study of existing company detection systems and a broader examination of collaborative intrusion detection systems. Based on this research, the most suitable topology for communication and distribution of nodes is determined, and the intrusion detection network is deployed accordingly. To address the aspect of zero trust, an environment that meets this requirement is identified. Blockchain technology emerges as a potential solution for creating such an environment with its distributed architecture and functionalities for traceability and interaction validation.

Apart from the network infrastructure, we have also developed and implemented a data-sharing and decision-making process that utilizes argumentative logic. By adopting this approach, we can capture the complexity and dynamism of multiple events and their relationships, leading to better detection and response to cyber threats. Our collaborative intrusion detection framework allows us to generate arguments from real-world detection alerts and use them in the argumentation process. This helps us to build an argumentation graph that captures the decision-making process, enabling us to respond more effectively to cyber threats.

In future work, we propose to enrich arguments with multi-stage attack indicators. By incorporating additional indicators and utilizing machine learning algorithms, the detection system can further enhance its ability to identify and respond to complex and multi-stage attacks, improving the overall security posture of large enterprises.

Given the continuous evolution of cyber threats, it is crucial to develop advanced collaborative intrusion detection solutions that leverage trust, argumentation logic, and enriched indicators

to counter the growing sophistication of attacks effectively. This research lays the foundation for such a solution and presents promising avenues for further development and improvement in collaborative intrusion detection.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ACRONYMS

IDS	Intrusion Detection System
CIDS	Collaborative Intrusion Detection System
DIDS	Distributed Intrusion Detection System
CIDN	Collaborative Intrusion Detection Network
AF	Argumentation Framework
AAF	Abstract Argumentation Framework
DLT	Distributed Ledger Technology

CHAPTER 1 INTRODUCTION

Over the years, the frequency and complexity of cyber-attacks have increased significantly [6]. In particular, in 2022, the number of attacks targeting an organization in a week increased by 28% compared to 2021 [7]. As attacks become more advanced and infrastructure becomes more complex and distributed, organizations must adapt by deploying effective defensive solutions to combat these sophisticated threats. Unfortunately, about 70% of business leaders today are uncertain about managing cybersecurity risks [8], and more than 50% of them believe that their IT departments lack the necessary sophistication to handle advanced cyber threats [9].

Early Intrusion Detection Systems (IDSs) [3] were typically standalone and only monitored a single system or network. The problem with this approach is that there is no communication or interaction between instances of a standalone IDS. Thus, it is ineffective in detecting sophisticated and widely distributed attacks such as Ransomware [10] and DDoS [3, 11]. A standalone IDS cannot establish connections between malicious events that occur in different locations because it lacks a global view [3]. Moreover, without the ability to monitor malicious activity globally, they may generate a significant number of false positives [12].

Collaborative intrusion detection systems have been proposed as an alternative since the late nineties [3]. This solution has proven to be of great use by offering the possibility to detect collaborated and distributed attacks. With the advancement of technology, numerous improved implementations have been proposed [3,13]. CIDS encourages a set of detectors to share and communicate necessary information with one another to provide a more efficient and comprehensive method for intrusion detection. In these systems, multiple agents or sensors must exchange information and make collective decisions to defend against cyber threats.

In the real world, every action we take involves a decision-making process. Managing daily tasks on our own might be relatively easy, but when it comes to more complex decision-making that involves interacting with others, it becomes difficult to avoid the need to work within a community. Reaching consensus is the ultimate goal of any multi-tier negotiation process, where multiple members agree on a particular decision. However, achieving consensus is not always easy, as it requires trust, which is a fundamental element of the negotiation process. To better understand this concept, let's take the example of a rescue team trying to evacuate victims trapped underground after an earthquake. Each member of the rescue team is assigned a specific site to assess and report on. If one of the members fails to inspect the

site correctly and provides false information, it can lead to further disaster. This situation usually arises when other members trust that the assigned task was executed correctly.

This is similar to what happens in a collaborative intrusion detection system, CIDS. These systems consist of multiple local intrusion detection nodes that work together to improve intrusion detection accuracy using collective knowledge and experience. However, there is a risk that some of these nodes may become unresponsive or malicious, leading to insider threats or system failure.

Despite their numerous benefits [3,14], CIDSs encounter several difficulties related to decision-making, trust management, and communication [2]. Effective decision-making mechanisms are required to deal with the uncertainty and diversity of the input data, ensuring the precision and timeliness of decisions in the face of contradictory or insufficient information. Furthermore, establishing and maintaining trust between participating detectors is essential to the system's effectiveness.

1.1 Problem Statement

Collaborative intrusion detection systems (CIDS) have been improved by trust-based solutions [2], but they may not fully utilize the potential of the collaborative environment. Existing solutions tend to focus on specific elements such as trust management or collaboration [15–18], which may lead to suboptimal results and reduce the efficacy of CIDS in dealing with complex cyber threats. This is because there is a lack of a holistic structure for informed decision-making that incorporates the system's collective knowledge and necessitates traceability and accountability.

To address concerns about transparency, accountability, and integrity, Distributed Ledger Technologies (DLTs) such as Blockchain have been incorporated into CIDS. DLTs and smart contracts exhibit potential in resolving the persistent challenge of centralized systems and the associated Single Point Of Failure (SPOF) concerns [13]. In some use cases, smart contracts can perform functions traditionally executed by central counterparties, such as making a decision and disseminating it.

However, using DLTs does not solve the trust problem amongst cooperating IDSs; it is still challenging to understand the context of a security warning and whether or not it represents a genuine risk. DLTs are only used to track the results of decision-making processes. On the other hand, a system with built-in reasoning capabilities to make decisions might improve transparency and traceability and provide explainability.

An argumentation framework is a structured approach to reasoning and decision-making. It

involves exchanging and evaluating arguments and counterarguments. It provides a framework for analyzing complex problems and facilitating collaboration among different stakeholders [19]. It provides numerous benefits including transparency, traceability, informed decision-making, and the ability to handle uncertainties and conflicts [20].

In the context of Collaborative Intrusion Detection Systems (CIDS), incorporating an argumentation framework can enhance the decision-making process. CIDS often involves multiple intrusion detection systems working together to detect and respond to cyber threats.By evaluating and combining arguments from different sources, the framework can improve the effectiveness of the system.

The argumentation framework provides transparency, allowing system administrators and analysts to understand the reasoning behind the decisions made by the CIDS. The traceability feature enables tracking and analysis of the flow of arguments and justifications. This promotes accountability and transparency, improving the overall performance of the system.

1.2 Research questions

The main objective of this research is to bolster the efficiency and reliability of Collaborative Intrusion Detection Systems (CIDS) by elevating the levels of transparency and trust in the decision-making process. We propose a novel solution for collaborative intrusion detection that we name XM-Guards, which employs an argumentation framework and blockchain technology to enhance transparency and trustworthiness in decision-making. Our approach helps to address issues of mutual distrust and context-based decision-making by allowing agents (IDSs) to evaluate the credibility of shared information and make informed decisions to counter potential cyber threats eventually.

The research questions that this thesis seeks to answer are as follows:

- How effective is the defined argument structure and formal argumentation framework in enhancing the decision-making process within Collaborative Intrusion Detection Systems?
- To what extent does the proof-of-concept implementation of XM-Guards validate the proposed solution's efficacy in a real-world scenario?

To answer our research questions, we followed a multi-stage approach. Firstly, we conducted a thorough review of existing literature and solutions in the Collaborative Intrusion Detection Systems (CIDS) field, focusing on trust-based solutions, formal argumentation frameworks, and the application of blockchain technology in cybersecurity. We also examined industrial

solutions that have attempted to address similar challenges in CIDS. This extensive review marked the second phase of our work. In the third phase, we developed XM-Guards, our innovative solution that combines an argumentation framework with blockchain technology for distributed deployment and consensus management. A crucial aspect of our work is the implementation of a proof-of-concept of XM-Guard. Our ultimate goal is to validate XM-Guards' effectiveness and applicability in real-world settings by testing it in synthetic scenarios.

1.3 Scope

Our main objective is to develop a new solution named XM-Guards, which utilizes an argumentation framework and blockchain technology for distributed deployment and consensus management. The system aims to enhance decision-making in Collaborative Intrusion Detection Systems (CIDS) by enabling agents (IDSs) to evaluate the credibility of shared information. Although we may provide insights on improving trust and transparency in CIDS, our primary focus is on our proposed solution. We have also tested a proof-of-concept implementation of XM-Guards with real-world data.

1.4 Thesis outline

We present our work in two parts. The first part provides the related work in two chapters. Chapter two delves into the concept of collaborative intrusion detection systems, where we review existing solutions and compare them to our objectives. In chapter two, we introduce. Collaborative Intrusion detection systems and the existent work in this field. In chapter three, we present Blockchain as a suitable infrastructure for our proposed solution and showcase various projects that utilize this technology for distributed decision-making. The second part is composed of three chapters. In chapter four, we explore the logic of argumentation and its relevance to our decision-making process. We also define our argumentation framework in the context of collaborative intrusion detection. Chapters five and six describe our research environment, data, and a thorough implementation analysis. We also express our experimentation process and results. Finally, we present concluding remarks and possible future work.

CHAPTER 2 LITTERATURE REVIEW PART 1 - COLLABORATIVE INTRUSION DETECTION SYSTEMS

Cybersecurity threats are becoming increasingly complex and sophisticated, which is why security professionals are paying more attention to CIDS. These systems are designed to consolidate the resources of multiple organizations to enhance threat detection and response capabilities.

In this chapter, we will begin by defining IDS and its components before moving on to explain the additional components that make up a CIDS, as well as its possible architecture and topologies. We will also provide an overview of commercially available CIDS solutions and academic works on the topic. To help consolidate your knowledge of CIDS, we will summarize the fundamental principles at the end of the chapter.

2.1 Basic Knowledge

In this section, we define intrusion detection systems and their components. Building on that, we introduce collaborative intrusion detection systems, their components, possible architectures, and topologies.

2.1.1 Intrusion Detection System (IDS)

An Intrusion Detection System (IDS) is a technology that monitors network or system activities to identify and respond to potential security incidents or policy violations. It aims to detect and alert administrators about unauthorized access, malicious activities, or policy violations in real time.

Types

Host and network-level intrusion detection systems (IDSs) are widely deployed to safeguard personal and organizational assets from cyberattacks.

IDSs can notify security administrators when a network is under attack [2].

There are two main types of detection systems [2, 3]: Signature-based and anomaly-based. Signature-based systems compare available signatures to observed events to identify known attacks. On the other hand, anomaly-based systems detect potentially malevolent events by comparing current profiles to predefined average profiles. A signature represents a pattern for a known attack and exploits, whereas an average profile can be generated by observing the target object for a predetermined time. Depending on the environment, IDSs can be classified as network-based or host-based.

Network-based IDSs analyze network traffic and its associated characteristics for anomalies. Host-based IDSs, on the other hand, analyze local system events and logs for malicious activity [2, 3].

Intrusion detection systems (IDS) can use both host and network-based passive monitoring and also integrate active monitoring through honeypots. Honeypots are cybersecurity mechanisms used to deceive and lure in potential attackers. They are designed to appear as legitimate and valuable targets to unauthorized users [2]. Honeypots can be a part of an IDS strategy and are often employed within fully operational networks and servers to serve as decoy systems [21], diverting criminal attention from the actual systems. By analyzing malicious activity within the honeypot, security professionals can identify and mitigate vulnerabilities. Through monitoring the activities and interactions within the honeypot, IDS can gather valuable insights into attack techniques, patterns, and trends. Including honeypots as part of an IDS strategy allows organizations to gather real-time information about potential threats and better understand attackers' methods and motivations. They provide valuable data to enhance intrusion detection capabilities and network security. A better visualization of the IDS type of monitoring is showcased in figure 2.1.

Coordinated and large-scale attacks such as stealthy searches, worm outbreaks, and distributed denial-of-service (DDoS) attacks that target online systems across multiple networks are some of the most dangerous attacks. However, detecting these types of attacks is challenging since isolated intrusion detection systems (IDS) typically monitor only a small portion of a network. Moreover, certain IDS may be better suited to detect specific categories of attacks, while others may be more effective in detecting other types of attacks [22].

Signature-based IDSs, for instance, are very effective at detecting known attacks based on a database of signatures. However, they may not be as effective at detecting new attacks. On the other hand, anomaly-based IDSs can detect a certain number of recent attacks, but their false positive rate for known attacks is typically relatively high. Anomaly detection architectures utilize either data mining or statistical analysis techniques to detect anomalies

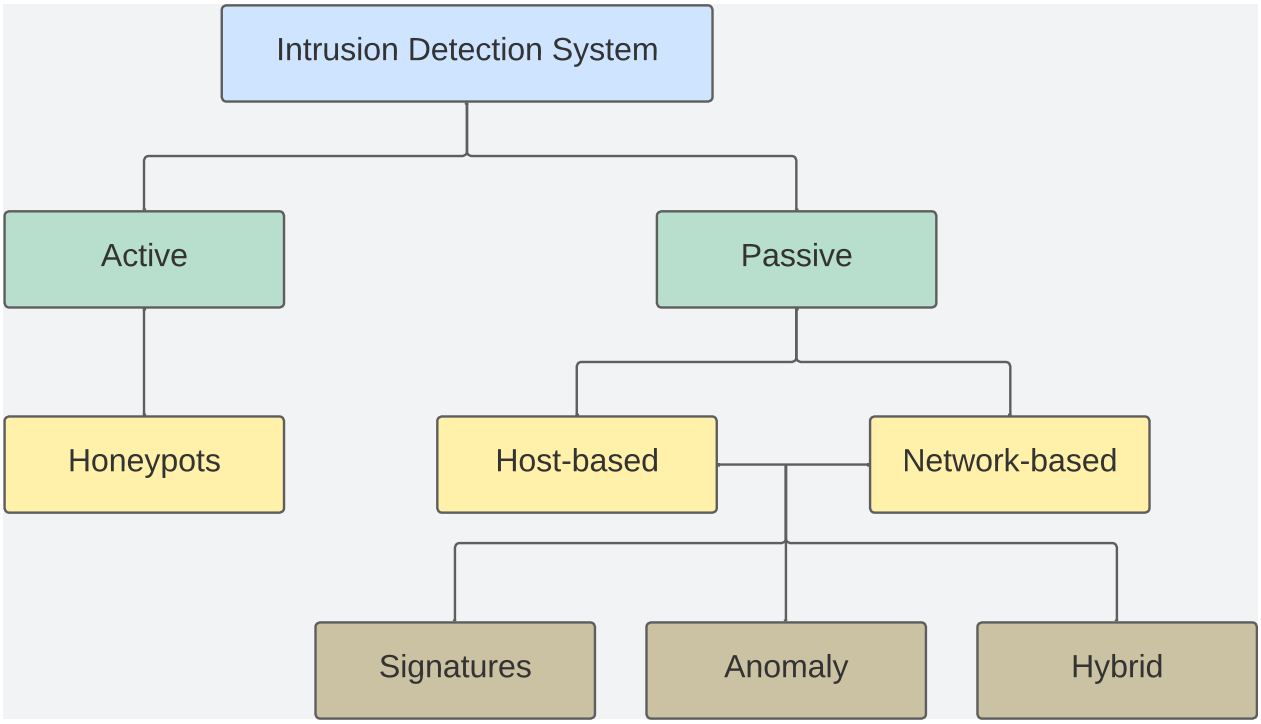

Figure 2.1 Monitoring classification in an intrusion detection system

in the monitored network traffic.

Components

A standard Intrusion Detection System performs many activities, including logging specific events and their corresponding information, alerting security administrators about serious incidents, and producing summary reports for analysts. Figure 2.2 depicts the fundamental constituents of an Intrusion Detection System (IDS), encompassing the sensor, management system, audit system, detection engine, database system, decision engine, and Graphical User Interface (GUI).

- The sensor serves as a hardware or device that allows the Intrusion Detection System (IDS) to monitor network activity and transmit this information to the system. In the domain of host-based intrusion detection systems (IDS), the sensing component is commonly referred to as an "agent."
- The management system is responsible for processing the data collected from various sensors and agents. In larger Intrusion Detection System (IDS) configurations, a specialized management server may be available as a separate hardware device or software

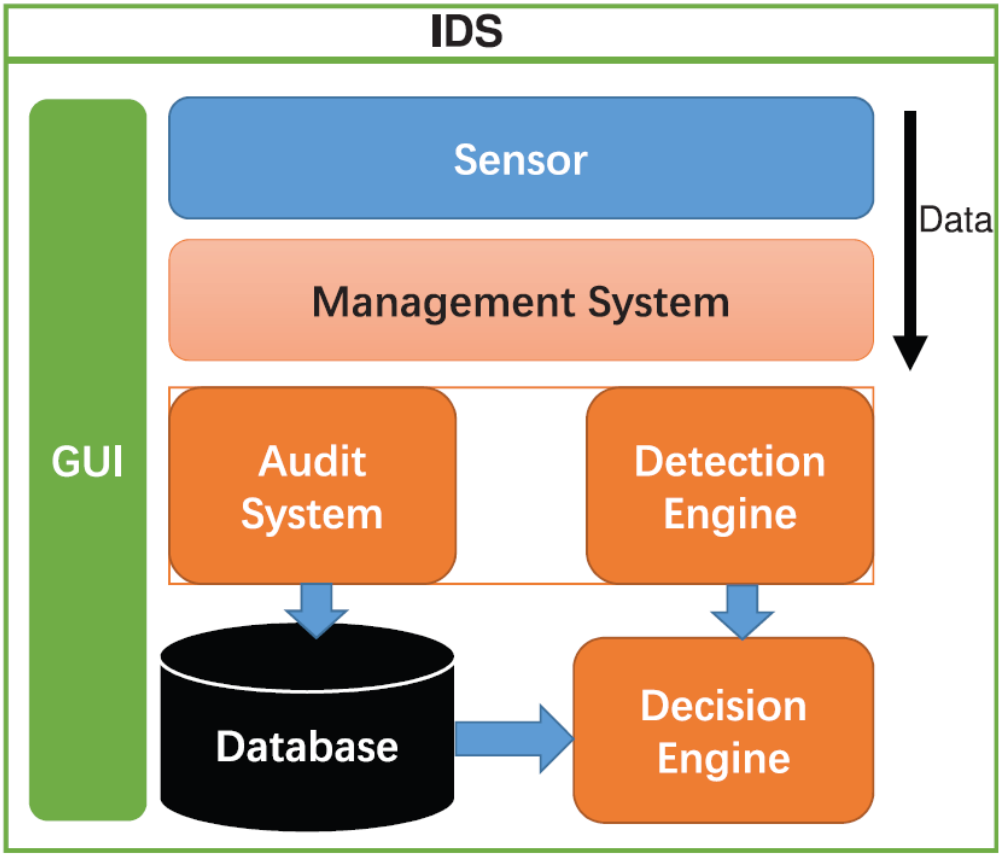

Figure 2.2 The major components of an IDS [2]

solution. Security managers can customize the system by implementing specific policies, enabling the pre-processing of data, and thus reducing the effort associated with detection.

- The audit system plays a critical role in maintaining accountability by recording and documenting user behaviors and system occurrences. These documents provide significant value in upholding the operating environment and serve as potential evidence in legal processes, should the need arise.
- The detection engine is essential in analyzing collected events from various sensors and agents. The engine can use both misuse and anomaly detection techniques, allowing it to identify notable deviations by comparing them with predetermined normal profiles.
- The database system is responsible for storing diverse information, including data collected by sensors and agents and information about the management system. Additionally, the system contains signatures, standard profiles, and security policies that the detection engine can access as needed.

- The decision engine generates warnings or alarms in response to harmful or notable occurrences, as determined by security administrators. To improve detection performance in large-scale networks, alarm correlation algorithms or specific metrics can be employed.
- The Graphical User Interface (GUI) provides a user-friendly interface to facilitate interactions, allowing security managers to efficiently oversee the Intrusion Detection System (IDS). This includes installing sensors or agents, establishing detection policies, and implementing software upgrades.

Managing and retaining sensitive data, such as host settings and known vulnerabilities, makes pivotal components a prime target for cyber-attacks. Therefore, it is crucial to prioritize safeguarding these elements by implementing targeted measures and employing supplementary security procedures.

Although IDSs come with different benefits, managing a large amount of data in real-time requires modern parallel hardware. Detecting anomalies in network traffic often requires time-consuming computational techniques such as data mining and statistical analysis, which are challenging to implement in real-time. Therefore, distributed environments such as parallel machines and cloud computing are essential to distribute the required computational load.

The above-mentioned factors have increased interest in distributed IDS, and various methodologies have been proposed.

2.1.2 Introduction to Collaborative Intrusion Detection System (CIDS)

As mentioned earlier, individual intrusion detection systems (IDSs) may not always be able to detect coordinated attacks on online systems across multiple networks. IDSs may struggle to identify innovative attacks such as ransomware, distributed denial-of-service (DDoS) attacks, and multi-stage attacks due to the rapid evolution of adversarial techniques. Additionally, they may generate a high number of false alarms and may not have the ability to monitor malicious activity globally, as stated in [2].

To address this issue and enhance detection capabilities, distributed intrusion detection systems (DIDSs) or collaborative intrusion detection networks (CIDNs) have been proposed and implemented. These systems encourage a set of detectors to communicate and exchange information, enabling a more comprehensive approach to threat detection and response.

Despite the various abbreviations such as DIDS, CIDS, or CIDN, which stand for distributed and collaborative intrusion detection, there is no distinction between them due to their

similar architecture and objectives. A collaborative intrusion detection system (CIDS) is a framework or system designed to improve the accuracy and efficiency of intrusion detection in a distributed environment. It involves cooperating and coordinating multiple components or entities within a network to collectively identify and respond to potential threats and attacks [2, 3].

Collaborative Intrusion Detection Systems (CIDS) are more effective at detecting coordinated attacks than isolated IDSs. This is because they can monitor multiple networks simultaneously and use various detection methods, such as signature-based and anomaly-based detection, by sharing and exchanging information. CIDSs can also distribute the computational load using distributed environments such as parallel machines and cloud computing, which enhances their ability to detect network traffic anomalies in real time and reduces false alarms. To better understand how CIDSs function, it is important to delve into the fundamental building blocks that enable their enhanced capabilities and performance.

2.1.3 CIDS Components

From an intuitive standpoint, it can be argued that a Collaborative Intrusion Detection System (CIDS) can potentially improve intrusion detection's effectiveness. This is achieved by allowing an Intrusion Detection System (IDS) node or detector to gather and share necessary information with other IDS nodes. Numerous collaborative systems have been proposed in the academic literature. Figure 2.3 illustrates a general architecture of CIDSs or CIDNs, encompassing a trust management module, collaboration module, communication module, and a standard IDS module.

- The IDS module encompasses the fundamental elements commonly found in an IDS, as seen in Figure 2.2. These components include a sensor, management system, audit system, detection engine, database system, decision engine, and other relevant components.
- The trust management module aims to assess the reliability of other intrusion detection system (IDS) nodes, also known as detectors, by using the information that is currently accessible. The significance of this module lies in the susceptibility of distributed systems to insider assaults. Therefore, several trust-based approaches can be employed to identify nodes that exhibit malicious behavior.
- The collaboration module has been built to facilitate the exchange of necessary data or information among different nodes. For instance, when there is insufficient data to make

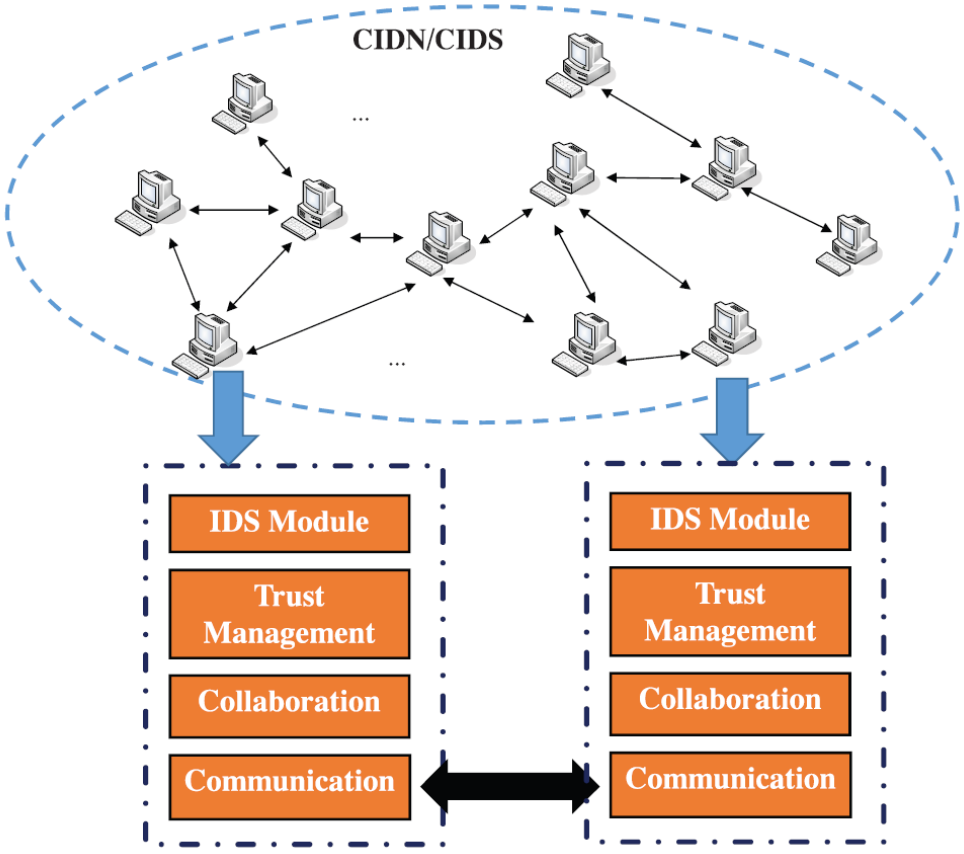

Figure 2.3 CIDS Components [2]

a conclusive judgment, the system can transmit an alert to its interconnected nodes for diagnosis. Subsequently, the system can gather input from the consulted nodes or be programmed to accept data from trustworthy nodes exclusively. This determination falls under the purview of the trust management module.

- The communication module primarily emphasizes the management of peer-to-peer (P2P) connections, namely the establishment of physical links with other IDS nodes within the collaborative network.

After exploring the fundamental components, we discussed the architecture and topology that underpin Collaborative Intrusion Detection Systems (CIDS). This will provide a comprehensive view of how these systems are structured and interact.

2.1.4 CIDS Architecture and Topology

When studying the topology and architecture of a collaborative intrusion system (CIDS), one important aspect to consider is the membership management provided. Membership management is responsible for ensuring that the CIDS monitoring overlay functions efficiently by coordinating with nearby sensors. This process can lead to the creation of static or dynamic overlays within the CIDS, which allow for the flexible addition and removal of monitors.

In the most basic scenario, the connections within the CIDS overlay are preset and static, which means that an administrator must be involved every time new components are integrated into the system. However, the CIDS overlay can also be configured dynamically using two different approaches. The first one involves utilizing a central server with a comprehensive system understanding, while the second one employs a membership management protocol that works independently at each monitor, relying solely on local knowledge.

As the membership management controls the overlay neighborhood of CIDS components, it can also impose a particular structure on it. Consequently, the resulting monitoring overlay may be centralized, hierarchical, or entirely distributed, as shown in figure 2.4. All monitors are directly connected to a central analysis device in a centralized CIDS. A hierarchical CIDS organizes all monitors in a hierarchy with a main analysis unit. Consequently, monitors at lesser tree levels report to those at higher levels. Moreover, hierarchical CIDSs include strategies that employ several super nodes. Distributed CIDSs prevent SPoFs by deploying monitors in a flat overlay containing no exposed components.

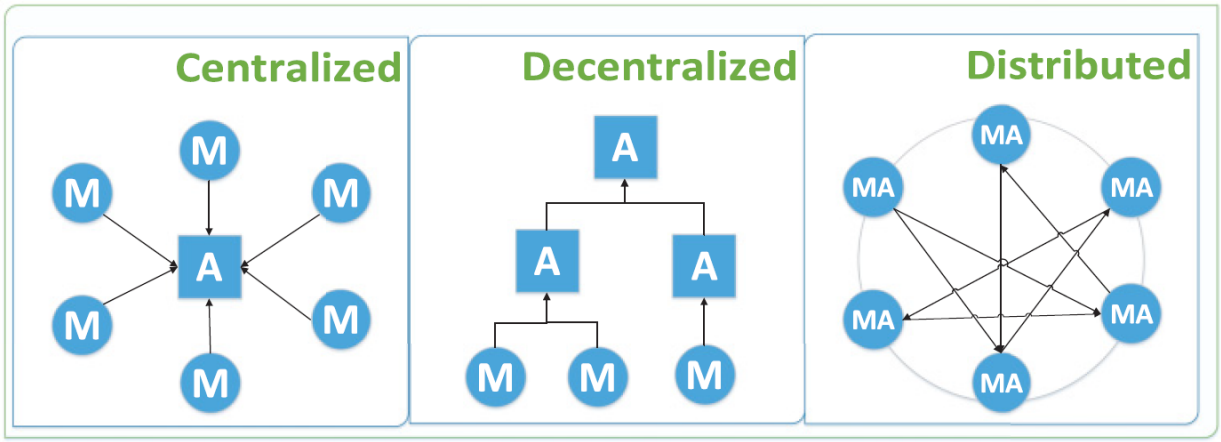

Figure 2.4 Possible architectures of CIDS [3]

In addition, the distribution of the CIDS components can be unstructured or structured.

In an unstructured overlay, no restrictions are imposed on how the peers are distributed or how they should exchange data. As this type of overlay offers flexibility, it does not provide an efficient way of storing and exchanging data between the neighbors. However, a structured overlay imposes a specific distribution on the participating sensors, for instance, by employing a Distributed Hash Table (DHT). The defined structure guarantees more efficient data dissemination and storage. On the other hand, it loses flexibility, and it is hard to ensure scalability in this topology.

2.1.5 Common Threats

CIDSs and CIDNs are specifically engineered to effectively identify intricate and geographically distributed assaults by promoting communication and interaction among interconnected nodes. However, it is essential to note that these detection systems might also be vulnerable to cyber-attacks. In this part, we will examine the prevailing external attacks and continue with the internal ones that raise risks to collaborative systems.

External Attacks

In the context of external attacks on CIDS, adversaries can detect the existence of monitoring nodes or collaborative detectors. They can then initiate evasion tactics and target specific components of the intrusion detection systems [2]. It is essential to highlight that modern attackers often employ automated tools to exploit multiple vulnerabilities simultaneously, making the task of detection challenging [23]. These attackers can utilize extensive scanning tools to penetrate a network to identify and exploit known software vulnerabilities. Once a vulnerability is identified, they can rapidly deploy malware or worm scripts to compromise many hosts simultaneously. Subsequently, these attackers may orchestrate large-scale DDoS attacks to disrupt the functionality of the targeted systems. Table 2.1 provides a set of external attacks with related case studies.

Table 2.1 Types of External Attacks on CIDS

Type of External Attack	**Description and Techniques**	**Examples/Case Studies**
Discovery/Probe	Discovery/Probe and active intruders may use probe-response hacking techniques to gather valuable information like detection results, sensor addresses, and device addresses. They may mark unique ports with distinct IP addresses to identify which systems are under monitoring.	Vasilomanolakis et al. [24]
Large-scale scanning	Large-scale scanning and attackers employ port scanning to learn network topology and identify vulnerable entities. Horizontal scans focus on a specific exploit across multiple entities, while vertical scans focus on all potential vulnerabilities of a single host. Block scans integrate both.	Birkinshaw et al. [25]
Worm/Malware	Worm/Malware and standalone malicious programs that replicate themselves to infect other systems. They may modify, delete, or replace files and inject additional malicious software.	Sapphire Worm [26], Stuxnet [27]
DDoS	DDoS and distributed Denial-of-Service attacks aim to compromise system availability by flooding the target with excessive internet traffic. They can target traffic, bandwidth, and application layers.	Vishwakarma et al. [28]

Internal Attacks

Insider threats are different from external attacks in that they come from individuals who have authorized access to the system. These individuals can be employees or contractors who misuse their access to compromise the confidentiality, integrity, or availability of specific assets. Insider threats can be categorized into two types: passive and active. Passive insiders have authorized access to resources but misuse this access to harm the system. Active insiders take a more aggressive approach, actively disseminating malware or worms to jeopardize the entire network.

Once an insider gains unauthorized control over a host within the network, they can launch various types of sophisticated attacks. They may establish a covert channel between the compromised internal host and an external entity to transfer sensitive information and spread malware undetected [2].

The compromised insider can then focus on exploiting other elements within the Collaborative Intrusion Detection Network (CIDN). This could involve revealing the locations of other IDS nodes, intercepting network traffic, exploiting protocol vulnerabilities, or disseminating false alarms. In more advanced scenarios, multiple compromised insiders could collaborate to execute intelligent attacks that are harder to detect, such as Distributed Denial-of-Service (DDoS) attacks or the removal of specific nodes from the network. There are different types of insider attacks: betrayal, Sybil, and collusion attacks, each with unique characteristics and impact. Table 2.2 provides internal attacks set with related case studies.

Table 2.2 Types of Internal Attacks on CIDS

Type of Internal Attack	**Description and Techniques**	**Examples/Case Studies**
Sybil Attack	An attacker forges multiple identities to control a fraction of a distributed system or network. Techniques include identity forgery and false information dissemination.	Affects routing algorithms like multipath or dispersity routing [29]. Affects data aggregation [30]. Workable in opportunistic networks (OppNets) [31].
Betrayal Attack	A highly reputed node becomes untrusted and penetrates other nodes. Techniques include gaining a high reputation before launching the attack.	Effective against Overlay IDS [32]. Degrades a trust management system [33].
Collusion Attack	Several malicious internal nodes work cooperatively to behave maliciously. Techniques include promoting each other and sending manipulated data.	Basic sleeper attack described by Fang et al. [34]. Pollution attack proposed by Meng et al. [35].

2.2 Commercial Solutions

When it comes to commercial offerings, details about their algorithms and infrastructures are often closely guarded secrets. Hence, we have sought to identify the most advanced Collaborative Intrusion Detection Systems (CIDS) available in the market and will present them in this section. There are various security tools recommended for managing and enhancing CIDS. Table 2.3 lists the identified products on the market and their description of how they fall under the collaborative intrusion detection systems category.

Table 2.3 Identified CIDS products

Product	**Corporation**	**Description**
Network Security Platform (NSP)	McAfee	While McAfee Network Security Platform (NSP) may not explicitly advertise itself as a collaborative IDS, it offers advanced security features, such as threat intelligence and real-time security updates, that integrate with other security solutions to provide a layered security approach.
QRadar SIEM	IBM	QRadar SIEM allows organizations to integrate and collaborate across different security tools, data sources, and teams. It provides a centralized view of the security posture, supports incident response workflows, facilitates threat intelligence sharing, and allows integration with other IBM and third-party security tools. Through collaboration and integration, QRadar SIEM enhances organizations' ability to detect and respond to security threats.
Deep Discovery Inspector	Trend Micro	Even though it is not clearly stated that Deep Discovery Inspector provides collaborative features, it is designed to work collaboratively with many Trend Micro solutions, including Deep Discovery Analyzer, Office Scan, and Apex One, improving threat detection and response capabilities.
Networks Next-Generation Firewall (NGFW)	Palo Alto	The NGFW is designed to work with other security tools and technologies, allowing for seamless collaboration and information sharing across the security ecosystem.
CrowdSec	CrowdSec	CrowdSec is a community-driven, collaborative security suite designed to detect and remediate malicious activities. It relies on a network effect where individuals and organizations contribute and share data, enhancing security effectiveness.

2.3 Previous Research Work

Collaborative Intrusion Detection Systems (CIDS) have undergone significant development over time as a research area [3]. Initially, trust was not a primary focus but as the complexity of distributed systems grew, the need for anticipating the reliability and behavior of different entities arose. Consequently, the concept of trust, which was first studied in social sciences, was later integrated into computer security. Trust in cryptographic situations is often associated with entities validated by a reliable third party. In addition, trust can be viewed as a numerical assessment of the level of certainty in accurately predicting the likelihood of an event's occurrence. Along with trust, reputation also plays a critical role in establishing trust-based connections across different organizations. Since some entities cannot observe the conduct of others directly, they depend on past experiences and shared knowledge to form judgments. The trust-centric approach has been instrumental in detecting attacks within distributed systems, particularly those originating from internal sources. In recent years, there has been a significant advancement in trust-based Collaborative Intrusion Detection Systems (CIDS), specifically focusing on identifying and mitigating insider attacks [2].

RepCIDN [12]

The Reputation-based Collaborative Intrusion Detection Network (CIDN) is a technology created to detect insider attacks by sharing alarm data and insights among nodes. The current methodology assesses the credibility of nodes within CIDN and filters out false alerts from malicious entities. The approach involves implementing a decentralized system, where specific nodes are designated as super-peers or super-nodes responsible for reputation management. These nodes are members of the Wise Committee (WC).

In the first stage, a Publication Service (PS) collects notifications from various nodes. Then, the WC evaluates the authenticity of the alarms and decides which ones to share with the public. The main goal of this technique is to effectively manage and minimize the occurrence of false alarms, thus reducing the burden on security managers. It's important to note that the members of the WC have high trust scores.

In relation to our research, this work emphasizes the significance of addressing insider threats and adopting a collaborative framework for threat detection. Trust and reputation management, which are central to RepCIDN, reflects an essential aspect of our research's focus on reliable network security. Furthermore, RepCIDN's decentralized structure, aimed at enhancing system resilience and reducing single points of failure, aligns with our goal of developing a robust and efficient intrusion detection system.

Li-ISCIDN [16]

In their actual implementations, Li et al. [16] observed that there could be variations in detection sensitivity among different detectors. Particular detectors may demonstrate increased sensitivity towards specific kinds of assaults. The authors established the notion of "intrusion sensitivity" to quantify the gap in capabilities among nodes. To allocate sensitivity values, the researchers devised a query component to evaluate detection sensitivity and assign the appropriate value afterward. Additionally, the researchers investigated the possibilities of utilizing machine learning techniques to optimize and simplify allocating value. The experimental findings show that using intrusion sensitivity may significantly improve the accuracy and effectiveness of detecting insider assaults.

How it relates to our research - Their introduction of "intrusion sensitivity" to address varying responsiveness of different detectors to specific threats parallels our approach to tailored cybersecurity measures in CIDS.

RevMatch [36]

The collaborative malware detection (CMD) technique known as RevMatch leverages its historical data to improve the accuracy of identifying dangerous applications. Functioning as a form of Collaborative Intrusion Detection Network (CIDN), this system facilitates the exchange of information and knowledge across individual nodes, enhancing the ability to detect and identify malware. The CMD system enhances its malware detection capabilities by aggregating data from several nodes or detectors. Significantly, the RevMatch method considers partial matches acceptable, even if a complete match is not obtained.

How it relates to our research - Its strategy of using aggregated data from various nodes to improve malware detection accuracy is akin to our method of harnessing collective network intelligence for threat identification.

T-CLAIDS [37]

The T-CLAIDS is an Intrusion Detection System (IDS) model based on trust-aware collaborative learning automata. It is designed mainly for Vehicular Ad Hoc Networks (VANETs). The system includes a Collaborative Trust Index (CTI) that adapts in response to operational outcomes, providing rewards or penalties contingent upon achieving or lacking desired results. The T-CLAIDS system collects essential data to identify anomalous activity within a designated geographical area. A node is identified as malicious when its CTI value exceeds a pre-established threshold.

How it relates to our research - Its use of a Collaborative Trust Index (CTI) that dynamically adapts based on operational outcomes parallels our interest in a responsive and adaptive trust-based system.

FACID [17]

The FACID model utilizes hypothesis testing techniques to improve detection accuracy and consolidate feedback. This design minimizes processing requirements and communication overhead in large-scale CIDN environments involving many cooperating nodes. By utilizing previous data, such as the frequencies of false alarms, the FACID system functions in two unique operational modes: sequential and non-sequential. The sequential mode involves querying each cooperating node sequentially until the work is completed, making it well-suited for situations with restricted resources. In contrast, the non-sequential mode can concurrently transmit requests to all nodes, rendering it suitable for networks of lower scales.

How it relates to our research - The similarity lies in the shared goal of optimizing system efficiency in large-scale network environments and minimizing communication overhead in environments with numerous cooperating nodes.

Meng-Bayesian [15, 38]

The study focused on identifying insider threats inside Medical Smartphone Networks (MSNs) through applying Bayesian inference. Integrating smartphones into healthcare systems has led to Mobile Social Networks (MSNs) emerging as a crucial platform inside healthcare organizations. Centralized components are essential in Mobile Sensor Networks (MSNs) since they play a vital role in data aggregation and enabling effective decision-making processes. This is particularly important due to the little IT expertise often possessed by most healthcare professionals. The team computed the reputation of each node, and afterward, an adaptive blacklist was created to include all harmful nodes. This approach received clearance from the security managers concerned. Following this, their research focus transitioned towards internet-connected gadgets. The researchers investigated the delicate equilibrium between security and usability by implementing an appropriate trust management paradigm. The researchers developed a CIDN framework to identify suspicious devices inside a healthcare-focused SDN environment to accomplish this objective. This framework employs a combination of behavioral profiling and the Bayesian model.

Their focus on implementing adaptive trust management aligns with our objectives of creating a dynamic security system tailored to specific organizational contexts. This alignment

underscores a shared emphasis on developing robust, adaptable, and efficient intrusion detection solutions.

Li-KNISCIDN [33]

The skills and knowledge of experts are crucial in improving Intrusion Detection Systems (IDSs) in practical situations. This particular skill set facilitates activities such as prioritizing and evaluating alarms. When considering the proficiency of security managers, it is essential to note that the detection effectiveness may vary depending on the exact detection nodes or engines being utilized. In their study, Li et al. [33] proposed the notion of "intrusion sensitivity" as a means to differentiate the levels of sensitivity exhibited by different intrusion detection systems (IDSs) in identifying specific threats. The researchers developed a trust-centric CIDN called KNISCIDN, which combines the KNN classifier and intrusion sensitivity to enhance the effectiveness of intrusion detection. The K-nearest neighbors (KNN) classifier provides a sensitivity rating to each node in the intrusion detection network (CIDN) with a high level of specificity. This assignment's focus revolves around utilizing the Euclidean distance measure.

How it relates to our research - Despite Li-KNISCIDN's focus on customizing sensitivity for specific nodes, the underlying principle of adapting detection strategies to diverse threat scenarios is a common goal.

TRAACK [39]

The Trust Based Adaptive Acknowledgment (TRAACK) Intrusion Detection System (IDS) model has been specifically developed to evaluate the reputation of nodes inside Wireless Sensor Networks (WSNs) by employing the Kalman filter. The authors classify a particular route into three levels of trust: low, medium, or high. The researchers implemented an Adaptive Acknowledgement (AACK) mechanism to reduce control overhead by selecting a route based on its level of trustworthiness. The Kalman filter is applied to ascertain the likelihood of a node or route exhibiting malicious behavior. Lights were allocated to different components to augment flexibility, enabling sensitivity level adjustment. The evaluations conducted on TRAACK have provided evidence supporting its efficacy in identifying and isolating deceptive nodes, thus maintaining a constant packet delivery ratio.

How it relates to our research - Its focus on dynamically evaluating trust levels and adapting network behavior accordingly aligns with our approach to creating a flexible and responsive security system.

Meng-FSCIDN [40]

In the forthcoming era of big data, managing trust in the face of substantial network traffic is a notable problem for Intrusion Detection Systems (IDS). The increase in network traffic can potentially overwhelm the detection engine, resulting in packet loss and reducing detection efficiency. In response to this issue, Meng et al. [40] proposed the implementation of traffic sampling techniques as a means to enhance the robustness of trust management in scenarios with significant traffic levels. The researchers developed a trust management system based on Bayesian principles for implementation in a hierarchical Wireless Sensor Network (WSN). The trustworthiness of individual sensor nodes in this configuration is assessed by selected cluster chiefs, using packet status as the basis for evaluation. The base station sets the dependability of these cluster heads. Expanding upon this notion, Meng et al. [41] proposed a refinement, suggesting the incorporation of traffic filtration and sample methods to boost the calculation of trust. Improving the sampling process and overall trust management may be achieved through refining and reducing unneeded network traffic. The empirical findings from their research confirmed the effectiveness of this integrated methodology.

How it relates to our research - Their strategy to handle increased network traffic without overwhelming the detection system aligns with our aim to ensure efficiency and accuracy in large-scale network monitoring.

BSDNFilter [42]

The authors Meng et al. [42] proposed BSDNFilter, a security mechanism based on intrusion detection system (IDS) principles designed explicitly for a software-defined networking (SDN) system connected with blockchain technology. The BSDN filter system has a dual methodology, including both blacklisting and whitelisting techniques. In this context, an SDN controller is utilized to mitigate harmful or unnecessary traffic within the network. The main goal of BSDNFilter is to provide a robust trust management system that can effectively handle the difficulties presented by large volumes of data and distributed denial-of-service (DDoS) assaults.

How it relates to our research - Its use of advanced technology, notably blockchain, for robust trust management and security, parallels our approach to leveraging cutting-edge solutions in network security.

Li-SpatialCorrelation [43]

Li et al. [43] introduced a unique trust management approach to improve the resilience of challenge-based CIDNs against a range of insider attacks. This method combines the ideas of challenge-based trust and behavioral trust. The assessment of behavioral trust involves evaluating the geographical correlation between nodes while considering characteristics such as forwarding latency, packet loss, and transmission rate. One notable characteristic of this methodology is its autonomy from past knowledge, indicating that it does not depend on pre-existing information on nodes' benign or malevolent nature.

How it relates to our research - This approach aligns with our research's emphasis on real-time analysis and adaptive trust assessment in network security, where the focus is on current network conditions and behaviors rather than solely on historical data.

Li-SamplingTrust [44]

Li et al. [44] conducted a study on Medical Smartphone Networks (MSNs). A unique traffic sample approach, enabled by blockchain technology, was proposed to enhance the robustness of trust management. This technique utilizes Bayesian inference, particularly in high-traffic settings. The proposed adaptive systematic sampling methodology adjusts the real-time traffic selection interval based on node repute variations. Given the inherent scarcity of information technology (IT) proficiency within healthcare facilities, implementing this efficient and simplified approach holds promise for reducing operating expenses within the healthcare industry.

How it relates to our research - Their use of blockchain for enhancing trust, coupled with applying Bayesian methods in high-traffic scenarios, aligns with our emphasis on incorporating advanced technologies and sophisticated analytical techniques for network security.

Table 2.4 summarizes the identified studies on CIDSs.

Table 2.4 Summary of IDS Models and Technologies

Model/Technology	**Description**	**Key Features/Methods**	**Reference**
RepCIDN	Collaborative IDS to identify insider attacks	Decentralized system, Wise Committee (WC) members, Publication Service (PS)	[12]
Li-ISCIDN	IDS focusing on variations in detection sensitivity	Notion of "intrusion sensitivity," query component, machine learning for value allocation	[16]
RevMatch	Collaborative malware detection (CMD) technique	Aggregates data from nodes, considers partial matches	[36]
T-CLAIDS	IDS for Vehicular Ad Hoc Networks (VANETs)	Collaborative Trust Index (CTI), adaptive response to operational outcomes	[37]
FACID	IDS using hypothesis testing methods	Sequential and non-sequential operational modes, focuses on reducing overhead	[17]
Meng-Bayesian	IDS for Medical Smartphone Networks (MSNs)	Bayesian inference, centralized components, adaptive blacklist	[15, 38]
Li-KNISCIDN	IDS focusing on expert knowledge and sensitivity	"Intrusion sensitivity," KNN classifier, Euclidean distance measure	[33]
TRAACK	IDS for Wireless Sensor Networks (WSNs)	Kalman filter, Adaptive Acknowledgement (AACK), three trust levels	[39]
Meng-FSCIDN	IDS for heavy traffic scenarios	Traffic sampling techniques, Bayesian-based trust management, hierarchical WSN	[40]
BSDNFilter	IDS for blockchain-based SDN	Blacklisting and whitelisting techniques, SDN controller for traffic management	[42]
Li-SpatialCorrelation	Hybrid trust management scheme	Challenge-based trust, behavioral trust, spatial correlation among nodes	[43]
Li-SamplingTrust	IDS for Medical Smartphone Networks (MSNs)	Blockchain-enabled adaptive traffic sampling, Bayesian inference	[44]

2.4 Conclusion

This chapter provides a comprehensive overview of Collaborative Intrusion Detection Systems (CIDS), an important area in cybersecurity due to the increasing complexity of cyber threats. We start by defining the fundamental concepts of Intrusion Detection Systems (IDS) and their components, which lays the groundwork for understanding the enhanced capabilities offered by CIDS.

Various aspects of CIDS are explored, including their components, architectures, topologies, and common threats. The integration of modules like trust management, collaboration, and communication in CIDS illustrates the complexity and multi-faceted nature of these systems. Additionally, we discuss different architectural models—centralized, hierarchical, and distributed—highlighting how they influence the efficiency and effectiveness of CIDS.

We examine commercial solutions, identifying products that embody the collaborative intrusion detection approach. While proprietary information often limits the depth of analysis, vital products aligning with CIDS principles are identified.

Previous academic research is reviewed, revealing a range of innovative approaches and methodologies. Each study contributes to the field by addressing specific challenges like insider threats, varying detection sensitivities, and efficient management of high network traffic. These studies offer insights into trust management, machine learning applications, blockchain integration, and advanced statistical methods.

The similarities between these studies and our research indicate a shared focus on developing responsive, adaptable, and efficient intrusion detection systems. The integration of advanced technologies, such as blockchain and Bayesian inference, and the emphasis on real-time analysis, flexibility, and scalability are common threads that run through these studies and our work. This collective body of research underlines the importance of collaborative approaches in addressing the evolving landscape of cybersecurity threats.

CHAPTER 3 LITTERATURE REVIEW PART 2 - BLOCKCHAIN APPLIED TO CIDS

The contemporary digital landscape is witnessing an escalation in both the frequency and complexity of cyber threats, as exemplified by notable incidents such as the Wannacry virus assault. This highlights the necessity for bolstered cybersecurity measures, particularly inside critical infrastructures. Although conventional security measures provide a certain level of safeguarding, they sometimes fail to encompass a holistic perspective of the complete network. The phenomenon above has resulted in the emergence of Collaborative Intrusion Detection Systems (CIDSs), which aim to offer a more comprehensive and cooperative strategy in the field of cybersecurity. Nevertheless, implementing collaboration in CIDSs has inherent obstacles, primarily focusing on establishing and maintaining trust between participating institutions.

In light of these issues, blockchain technology is a prospective remedy. Initially designed for cryptocurrencies, blockchain provides a decentralized and irreversible record that enables transparency and consensus without central authority. The fundamental characteristics of this system are the requirements of CIDSs. The decentralization of blockchain technology bolsters trust in CIDSs by mitigating the potential for undue influence or dominance exerted by a singular entity. The immutability of the system guarantees the integrity of the data. At the same time, its consensus processes assure the system's robustness, positioning it as a possible solution for tackling the issues faced by CIDSs.

3.1 Basic knowledge

This section defines the main concepts related to the blockchain technology.

3.1.1 Blockchain

Blockchain [45] is an innovative instance of Distributed Ledger Technology (DLT) that emerged in 2008 alongside the introduction of Bitcoin. Distributed Ledger Technology (DLT) is a decentralized network of nodes that collectively maintain a shared protocol to manage a distributed ledger. In blockchain technology, the ledger is structured as a series of blocks organized sequentially and interconnected to their respective preceding blocks [45]. The process above generates a chain of data that is resistant to tampering and cryptographically secure. This results in a reliable mechanism for users in a distributed environment, eliminat-

ing the need for intermediaries from third-party entities. Figure 3.1 illustrates an example of a blockchain structure.

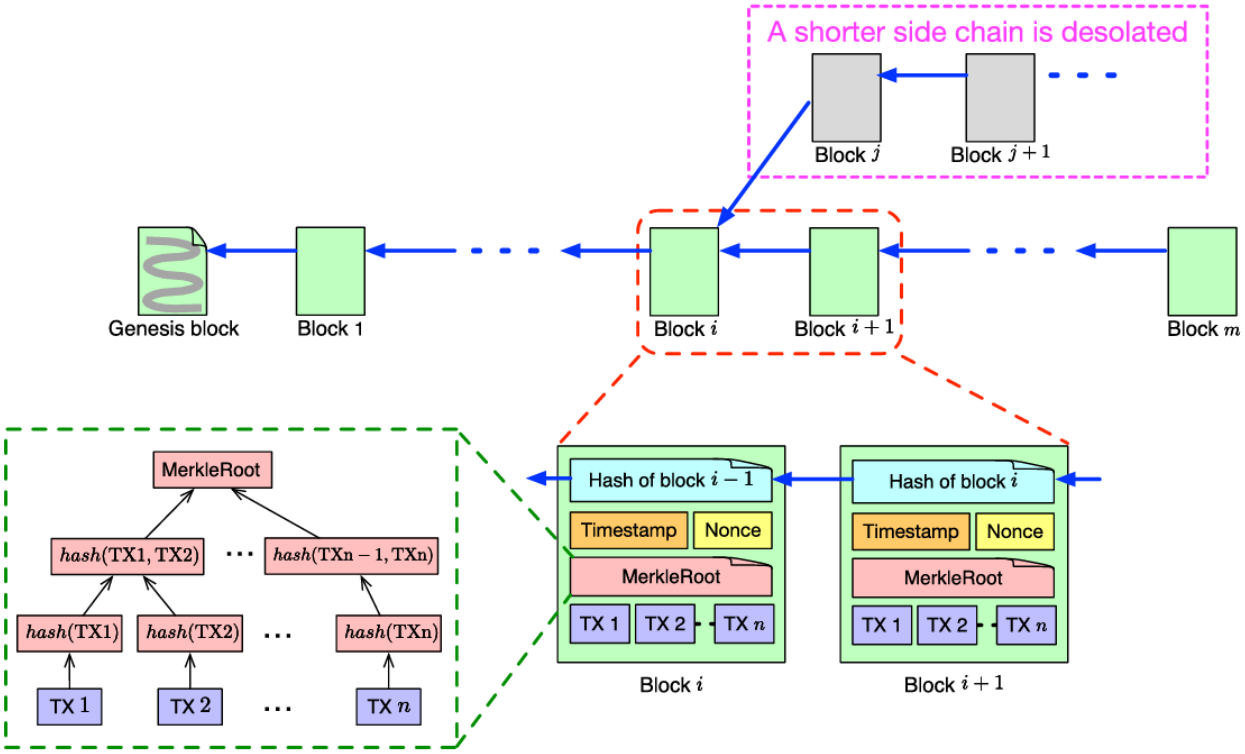

Figure 3.1 Blockchain structure [4]

Components

The foundational elements upon which the blockchain infrastructure and network functions are as follows [46]:

- **Asymmetric encryption** - The security and integrity of blockchain technology transactions primarily depend on the concepts of asymmetric key cryptography. In the blockchain network, every user possesses a digital wallet that functions similarly to a traditional bank account. This wallet is secured through the use of a distinct private key. Using this private key is crucial in producing digital signatures for transactions, guaranteeing their authenticity, and verifying that the legitimate owner of the wallet has started them. Similarly, it is worth noting that every wallet possesses a public key that serves as a publicly accessible address inside the network. In the context of cryptographic systems, it is common practice for the public key to be openly disseminated. In contrast, the private key is kept confidential and only known by the individual who

owns the corresponding wallet. Implementing a dual-key system guarantees that transactions may be subjected to public verification, but their initiation is restricted solely to the legitimate owner. Consequently, this security mechanism establishes a strong level of protection.

- **Transactions** - The blockchain network's fundamental principle revolves around transactions. These entities can be seen as records of transactions or transfers, which are recorded as files and disseminated over the whole network for verification. In cryptocurrencies such as Bitcoin, a transaction transfers digital money from one user to another. Every node inside the network retains a duplicate of the blockchain, which serves as a ledger documenting a sequential record of all previous transactions. The state of the blockchain undergoes evolution with the occurrence of each new transaction. Due to the decentralized structure of the network and the considerable magnitude of transactions, it is crucial to validate each transaction to ascertain its legitimacy and deter fraudulent activity.

- **Consensus Mechanism** - The decentralized structure of blockchain networks poses a distinct issue: the need to establish consensus across all network nodes on a singular version of truth, particularly in the absence of a central governing entity. The consensus process is employed in this context. The protocol guarantees consensus among all network nodes on the legitimacy of transactions and the current state of the blockchain. The Proof of Work (PoW) consensus technique is widely recognized, particularly within Bitcoin. Within the context of Proof of Work (PoW), nodes, also known as miners, engage in a competitive process to solve intricate cryptographic challenges. The individual who successfully solves the challenge is granted the privilege to append a new block to the blockchain and is duly compensated for their endeavor. The process above validates and records transactions while facilitating the creation of new money inside the Bitcoin system. The consensus mechanism is responsible for maintaining the integrity and security of the blockchain by safeguarding it against any potential tampering or harmful attacks.

Categories

Blockchain systems can be classified into three different categories:

- **Public Blockchain** - A public blockchain may be likened to a readily available and readable transparent ledger by many individuals. The system operates on democratic principles, allowing individuals from all backgrounds to partake, conduct trades, and

engage in mining. Referred to as "permissionless," this blockchain variant does not impose gatekeeping or operational restrictions. All participants, irrespective of their identities or intentions, possess the capability to start and validate transactions, explore the historical records of the blockchain, and potentially participate in the mining process to earn incentives. The whole system operates on the principle of decentralization, wherein each node or participant possesses a synced copy of the blockchain. In conjunction with consensus measures like proof of work, the system's decentralization safeguards its security and integrity, even in the event of open accessibility.

- **Private Blockchain** - Unlike its publicly accessible counterpart, a private blockchain might be considered an exclusive association limited to its members. This design is intended for a distinct cohort, which may be confined to a singular institution or extend to many organizations. In this context, the mining process is commonly overseen by a particular entity or a limited cohort of personnel. The "permissionned" characteristic entails that one cannot gain access without an invitation or consent. Although the system maintains a decentralized network structure among its constituents, it does not possess complete openness. Each node operates collectively in this system to preserve the ledger's integrity and achieve consensus. However, there exist limitations on the individuals authorized to add entries or "writes" to the blockchain.

- **Consortium Blockchain** - The consortium blockchain is characterized as a hybrid approach that operates at the intersection of public and private blockchains. Instead of a centralized authority or universal control, a consortium of predetermined nodes or organizations collectively assume governance. The designated entities are responsible for establishing the game's regulations and determining the eligibility of participants to join the network and engage in mining activities. In the context of block validation, it is essential to note that the process is not arbitrary. A block is considered legitimate only if it receives approval from predetermined nodes. The present system has a semi-centralized structure. Although it does not possess the same level of openness as public blockchains, it does not exhibit the same degree of exclusivity as private blockchains. The determination of the extent of public access for reading or writing to the blockchain is made by the consortium. Nevertheless, it is essential to acknowledge that the semi-centralized character of this system may render it susceptible to vulnerability if a majority of the consortium opts to modify the status of the blockchain.

3.1.2 Smart Contracts

The notion of "smart contracts" represents a progression in the realm of blockchain technology [47]. Self-executing contracts are essentially agreements whose terms are directly encoded into the software code. The contracts are stored within the blockchain, ensuring that each action or stipulation is recorded as an unchangeable transaction. This ensures the consistency of access control and contract enforcement. Developers can set specific access permissions for each contract function. When a smart contract condition is met, the associated function is triggered in a predetermined manner. For instance, if Alice and Bob have a contract in which Bob is subject to a penalty for any breach, the penalty is promptly deducted from Bob's deposit the instant he violates the terms [4]. This is a testament to how smart contracts are executed automatically upon fulfilling predetermined conditions, thereby broadening the scope of blockchain technology beyond mere transactional information.

The life cycle of a smart contract encompasses four distinct phases [4]:

- **Creation phase** - This phase begins with multiple parties discussing and negotiating the contract's terms, rights, and restrictions. After several discussions, an agreement is achieved. Legal specialists aid in crafting the initial contract, which is then translated into a smart contract using computer languages by software engineers. This translation incorporates both logic-based and declarative rule languages. The creation process is iterative, with multiple rounds of negotiations and revisions, and requires collaboration between stakeholders, legal specialists, and software developers.
- **Deployment Phase** - Smart contracts are deployed on blockchain platforms after validation. Due to the immutability of blockchains, once a contract has been deposited, it cannot be modified. Any modifications require the establishment of a new contract. Once deployed, the contract is accessible to all stakeholders via the blockchain. In addition, the digital assets of the parties engaged in the smart contract are protected by suspending the associated digital wallets, thereby temporarily halting any coin transfers related to the contract. During this phase, parties are identified through their digital accounts.
- **Execution Phase** - The smart contract clauses are continuously monitored and evaluated after deployment. When certain conditions (such as the receipt of a product) are met, the relevant contract procedures or functions are automatically triggered and executed. A smart contract consists of multiple interconnected declarative statements. When a particular condition is completed, the corresponding statement is executed, resulting in a transaction that blockchain miners subsequently validate. These validated

transactions and subsequent state updates are then added to the blockchain.

- **Completion Phase** - All parties' states are updated when a smart contract is executed. The blockchain stores the transactions that transpired during the contract's execution and the updated state. Then, digital assets are transferred between parties, such as when a client transfers funds to a seller. Once this transfer is complete, the digital assets of the parties involved are unsealed, signifying the conclusion of the life cycle of the smart contract.

While the concept of smart contracts is transformative, the Ethereum platform has brought them to the forefront of blockchain technology.

3.1.3 Ethereum

The launch of Ethereum [47] 2013 is considered a significant milestone in developing blockchain technology. The use of smart contracts is expanded, thereby enabling programmable transactions. Ethereum provides a comprehensive infrastructure for developing and implementing decentralized applications (dApps) [47]. This platform empowers developers to create and execute smart contracts and dApps with minimal downtime, fraud, or interference risk. The Ethereum platform demonstrates the vast capabilities of blockchain technology, providing a diverse and comprehensive set of applications.

Ethereum uses several consensus mechanisms in both public and private contexts. The existing consensus mechanism used on the public Ethereum blockchain is referred to as Proof of Stake (PoS), more particularly recognized as Ethereum 2.0. The objective is to substitute the preceding Proof of Work (PoW) method employed in Ethereum 1.0 [48]. Proof of Stake (PoS) is a consensus mechanism that depends on the participation of validators who own and immobilize a certain quantity of bitcoin as a stake to safeguard the network and authenticate transactions. The technique above exhibits a reduction in energy consumption and an enhancement in scalability compared to the Proof of Work (PoW) mechanism [48].

Consensus methods inside private Ethereum blockchains may vary depending on the implementation or platform employed. Several commonly used consensus algorithms in private Ethereum networks include [49]:

- **Proof of Authority (PoA)**: This method is predicated upon a designated group of authorized entities responsible for verifying transactions and ensuring network security. In many cases, individuals in positions of authority are chosen in advance and held accountable for their activities, resulting in centralization and effectiveness. Clique

algorithm [50] is seen as a derivative of the Proof of Authority (PoA) method. In a network structured on cliques, a predetermined group of approved authorities or validators sequentially create and validate blocks. The network's authority nodes must be recognized and trusted participants, as they can verify transactions and ensure the network's security. The Clique consensus mechanism presents a direct and practical approach to achieving consensus inside private Ethereum networks, whereby the establishment of trust among players is a fundamental aspect. Clique enables these networks to attain rapid block production durations while upholding security and data integrity. Due to its pre-established validators, the clique offers enhanced governance and network control compared to alternative consensus algorithms.

- **Raft Consensus:** Raft is a consensus protocol that offers strong consistency and fault-tolerance in private Ethereum networks. It elects a leader as an authoritative node to validate and order transactions. Raft is known for its simplicity and ease of implementation.

- **Byzantine Fault Tolerance (BFT):** BFT consensus algorithms, such as Practical Byzantine Fault Tolerance (PBFT), are designed to withstand malicious behavior and maintain consensus in the presence of faulty nodes. BFT algorithms are commonly used in private Ethereum networks with known trusted participants.

3.2 Previous Research Work

The increasing prominence of blockchain technology in cybersecurity has attracted considerable interest because of its ability to facilitate collaboration. The trust and transparency inherent in the decentralized structure of blockchain make it an intriguing technology for exploration in many collaborative settings, particularly in intrusion detection. The initial part examines the function of blockchain technology in establishing trust within Collaborative Intrusion Detection Systems (CIDSs). The subsequent part provides a more comprehensive analysis of the role of blockchains as a fundamental instrument for facilitating cooperation.

The enthusiasm surrounding blockchain technology has resulted in widespread adoption across several industries. These applications require separate blockchain implementations designed to cater to a particular use case and give distinct characteristics. Several solutions have been devised to augment privacy, enabling several entities to interact while ensuring the total confidentiality of stored data [51]. Some individuals use cryptographic identification systems to guarantee complete anonymity and lack of connection between transactions [52]. The energy sector has used the potential of blockchain technology to establish a peer-to-peer

market, allowing computers to independently engage in energy trading according to predetermined criteria [53,54]. Azaria et al. established a decentralized record management system in the medical arena, employing blockchains to uphold data integrity and impose access control regulations.

The current proliferation of blockchain applications is particularly noteworthy within the field of cybersecurity, with a specific emphasis on the Internet of Things (IoT) and software-defined networking (SDN). Due to the decentralized nature of IoT devices, they are vulnerable to a wide range of cybersecurity risks. Collaborative intrusion detection systems (CIDSs) have been developed as a viable option, enabling the sharing of critical data across detection nodes. This data exchange augments the detection capabilities when facing complex assaults. A noteworthy strategy that has attracted much interest in this particular setting is the challenge-based trust mechanism, which assesses the reliability of nodes using challenge-response exchanges. Numerous studies have provided evidence of the robustness of challenge-based collaborative intrusion detection networks (CIDNs) in countering conventional insider threats. However, apprehensions over their vulnerability to sophisticated insider assaults continue to exist. One notable development, as emphasized in current scholarly investigations, involves examining blockchain technology as a potential tool for augmenting the resilience of challenge-based collaborative intrusion detection networks (CIDNs) [55]. These studies propose blockchain technology can enhance trust management skills and improve alert aggregation procedures. This presents a significant opportunity to strengthen these systems against traditional and sophisticated attacks [55].

Concurrently, the emergence of Software-Defined Networking (SDN), a technology that facilitates network administration by separating the controller plane from the forwarding plane, brings forth its distinct array of difficulties [56]. This architecture, while efficient, introduces vulnerabilities, especially from insider threats where attackers can operate maliciously within the network. In light of these problems, there is a growing emphasis on the integration of trust management with blockchain technology to enhance the security and reliability of Cyber Intrusion Detection Systems (CIDSs) inside Software-Defined Networking (SDN) systems [56]. Trust management is a crucial process that aims to assess the dependability and credibility of individual nodes inside a network. Simultaneously, blockchain technology guarantees safe communication by eliminating the necessity for a trusted middleman, therefore maintaining the integrity of exchanged data [13, 56].

In the previous conversations, we explored the many uses of blockchain technology in several fields, highlighting its potential to bring about significant changes in terms of trust, privacy, and cooperation. The adaptability of blockchain technology is visible in several ap-

plications, ranging from its contribution to strengthening Collaborative Intrusion Detection Systems (CIDSs) to its integration within the Internet of Things (IoT) and Software-Defined Networking (SDN). To comprehensively cover the diverse range of applications and offer a systematic perspective, a summary is presented in the table 3.1 provided below. The text classifies the key domains or fields in which blockchain technology has been used, provides detailed explanations or descriptions of each domain, and refers to relevant sources that offer a more comprehensive understanding of each application.

Table 3.1 Summary of Blockchain Applications in Various Domains

Domain/Area	**Description/Contribution**	**Reference(s)**
Blockchain in CIDSs	Establishing trust within Collaborative Intrusion Detection Systems	[13], [57]
Blockchain Implementations	Augment privacy, cryptographic identification, peer-to-peer energy market, decentralized record management in medical	[51], [52], [53], [54]
IoT and CIDSs	Challenge-based trust mechanism in CIDNs, resilience against insider threats	[55]
SDN	Integration of trust management with blockchain in CIDSs, addressing vulnerabilities from insider threats	[56], [13]

3.3 Conclusion

This chapter starts by outlining the fundamental aspects of blockchain, such as its structural components, network types, and its significance in securing transactions using asymmetric encryption. It explains how blockchain creates a secure, decentralized ledger system that resists tampering and promotes transparency, crucial for enhancing trust in CIDSs.

Additionally, the concept of smart contracts in the blockchain is introduced, automating transactional agreements and replacing intermediaries or oversight. The chapter showcases how the Ethereum platform provides an excellent example of the versatility of blockchain in not only financial transactions but also in decentralized applications (dApps) and smart contracts.

Moreover, the chapter explores the application of blockchain in different domains, emphasizing its significance in CIDSs, IoT, and SDN. In CIDSs, blockchain enhances trust and

transparency, improving collaborative efforts in intrusion detection. Within the IoT and SDN domains, blockchain addresses some of the unique challenges of decentralized structures and insider threats for enhancing security.

CHAPTER 4 ARGUMENTATION FRAMEWORK APPLIED TO COLLABORATIVE INTRUSION DETECTION

The study of argumentation analyses how people construct arguments and how those structures might be replicated in a computational setting. Dung introduced it as a specific form of logic programming with negation as a failure that ensured the generation of meta-interpreters for argumentation systems by illustrating them as arguments with attack relations between them [58]. There is also widespread agreement that abstract argumentation has significantly impacted the development of AI. It offers a detailed and structured approach to reasoning and decision-making that facilitates handling conflicting, incomplete, or inconsistent information. It can give various ways, in terms of conversation or evidence, for explaining the reasoning behind a choice that has been reached [59]. Indeed, one advantage of argumentation is its capacity to visually depict decisions through predetermined attack properties, thereby demonstrating the route taken to reach the successful decision. The present concept will be expounded upon by providing precise definitions for an argumentation framework in the context of collaborative intrusion detection. We begin by generalizing the argumentation frameworks originally introduced by Dung (1995) [58]. These frameworks are directed graphs wherein the nodes represent arguments, and the arrows signify the attack relation.

4.1 Abstract Argumentation Framework

An abstract argumentation framework is a formalized conceptual framework employed to represent and examine arguments and their interconnections in a systematic manner. Utilizing a logical framework facilitates analyzing intricate arguments and their interconnections.

In an abstract argumentation framework, arguments are regarded as discrete information units or assertions that other arguments may substantiate or refute. The framework establishes a set of norms and criteria to assess arguments' validity or efficacy, considering their interconnections with other arguments.

4.1.1 Definitions

Definition 4.1.1 (Argumentation Framework (AF) (Dung 1995)). *An Argumentation Framework, denoted as* $AF = (AR, attacks)$*, is comprised of:*

- *A set AR of arguments, and*

- *A binary relation: attacks* $\subset (AR \times AR)$.

For any two arguments A and B in AR, the notation: $(A, B) \in attacks$*, signifies that argument A launches an attack against argument B.*

Given an Argumentation Framework $AF = (AR, attacks)$, we can define the notions of conflict-free and admissible sets as follows:

Definition 4.1.2 (Conflict-free and admissible Argument Sets (Dung 1995) [58])**.** *Given a set* $S \subseteq AR$ *of arguments:*

- ***Conflict-free Set****: S is said to be* conflict-free *if there are no arguments* $A, B \in S$ *such that* $(A, B) \in attacks$.
- ***Admissible Set****:*
  - *An argument* $A \in AR$ *is* acceptable *with respect to S if and only if for each argument* $B \in AR$ *such that* $(B, A) \in attacks$*, there exists an argument* $C \in S$ *such that* $(C, B) \in attacks$*. In other words, if B attacks A, then B is attacked by an argument in S.*
  - *A conflict-free set S is* admissible *if and only if each argument in S is acceptable with respect to S.*

As per Dung's study, extensions related to an argumentation framework are defined as follows:

Definition 4.1.3 (Extensions (Dung, 1995a) [58])**.** *Let* $AF = (AR, attacks)$ *be an Argumentation Framework. A set* $E \subseteq AR$ *is:*

- ***Complete Extension:*** *E is a complete extension if and only if E is admissible and* $\forall a \in AR$ *that are acceptable with respect to E, belong to E.*
- ***Grounded:*** *E is a grounded extension if it is the minimum (concerning set inclusion) complete extension.*
- ***Preferred:*** *E is a preferred extension if and only if it is the maximal (concerning set inclusion) complete extension.*
- ***Stable:*** *E is a stable extension if it is conflict-free and attacks each argument that does not belong to E.*

4.1.2 Related Work

In this work, we shall go into the process of creating a decision-making system for a collaborative intrusion detection environment. The system uses smart contracts, Blockchain technology, and logical argumentation. Therefore, we will provide a comprehensive overview of research covering this topic from various perspectives.

To our knowledge, no substantial work has been done incorporating argumentation into collaborative intrusion detection. Argumentation logic has been previously used in cybersecurity-related works. Rowe et al. [60] did a preliminary study on using argumentation logic to facilitate efficient decision-making in the context of security administration tasks. While this approach was innovative, it was limited to administrative contexts and did not explore intrusion detection. The work of Bouyahia et al. [61] presents an argumentative logic framework to identify and select the optimal countermeasure during an intrusion detection procedure while taking into account the particular attack and its contextual factors to prevent any adverse consequences on the system. This work was significant in considering specific attacks and contextual factors. However, it did not integrate a decentralized architecture, such as blockchain, which could enhance trust in the system. Bandara et al. [62] proposed using argumentation-based preference reasoning to translate security requirements into firewall configuration rules, automatically analyze inconsistencies, and generate firewall configurations. Argumentation logic was also employed in other fields. Karafili et al., [63] applied argumentation logic in data processing and management to emphasize the importance of data-sharing agreements (DSAs) and quality attributes.

More recent work on argumentation-based decision-making in the financial field was led by Yu et al. [64], which specifically integrated the argumentation using smart contracts and Blockchain technology in the fund management process, where trust was crucial. This work demonstrated the potential of argumentation in trust-sensitive domains but did not extend to intrusion detection. Solutions about trust management in collaborative intrusion detection systems were proposed but with different approaches than argumentation, [2]. Gil et al. [12] tackled the trust management problem by evaluating detectors based on reputation and trust values. Kumar et al. [37] also focused on managing trust by proposing a collaborative trust mechanism that uses vehicular communication to establish and update vehicle trust levels. Adding intrusion sensitivity in the solution of Li et al. [16] increases the efficiency and accuracy of identifying insider attacks, answering the challenge of robustness and trust. Other collaborative approaches, introduced by the studies of Fung et al. [36] and Li et al. [33], use historical data and machine learning techniques to answer the need for accurate decision-making by improving detection results. Meng et al. are [15] emphasize Bayesian inference,

and Li et al.'s [18] hybrid trust management approach contributes to robustness as a related concept to trust. Both studies of Fung et al. [17] and Meng et al. [65], which respectively use hypothesis testing and blockchain-based trust management technique that protects communication and consensus across nodes, address communication and computational overhead related to CIDS.

However, all previously cited systems fail to articulate the reasoning behind decisions, making it challenging to understand or validate them. Alexopoulos et al. [13], in their work, explore the intersection of Blockchain technology and CIDS by the integration of Blockchain as a means to enhance aspects like trust, accountability, and consensus within said systems, but no concrete implementation was proposed. Previous research in the domain of intrusion detection has made notable progress in the areas of trust management, decision-making, and the incorporation of Blockchain technology. Although these contributions hold significant value, they frequently lack a mechanism for delivering informative and comprehensible decisions.

Based on the research mentioned above, it becomes apparent that although the logic of argumentation has been examined in several cybersecurity scenarios, its use in collaborative intrusion detection remains relatively unexplored. The table 4.1 presented below provides a concise overview of prominent research studies that have explored the field of argumentation logic in cybersecurity. The purpose of each entry in the table is to clearly explain the significant emphasis or contribution made by the authors, the specific field of their study, any incorporation of modern technology, and any notable remarks or features of their work. Table 4.1 aims to provide a comprehensive reference that assists in identifying areas where current research is lacking and suggests directions for future investigation.

Table 4.1 Summary of Research Works on Argumentation Logic

Author(s)	**Main Focus/ Contribution**	**Context/ Domain**	**Integration with Blockchain**	**Notable Remarks**
Rowe et al. [60]	Use of argumentation logic for decision-making in security administration tasks	Security administration	No	Limited to administrative contexts
Continued on next page				

Table 4.1 – continued from previous page

Author(s)	Main Focus/Contribution	Context/ Domain	Integration with Blockchain	Notable Remarks
Bouyahia et al. [61]	Argumentative logic framework for optimal countermeasure during intrusion detection	Intrusion detection	No	Considers known attacks and contextual factors
Bandara et al. [62]	Translate security requirements into firewall configuration rules using argumentation-based preference reasoning	Firewall configurations	No	Automatic analysis of inconsistencies
Karafili et al. [63]	Application of argumentation logic in data processing and management	Data processing and management	No	Emphasizes data sharing agreements (DSAs) and quality attributes
Yu et al. [64]	Argumentation-based decision-making in the financial field using smart contract and Blockchain	Financial domain	Yes	Demonstrates potential of argumentation in trust-sensitive domains
Gil et al. [12]	Trust management by evaluating detectors based on reputation and trust values	Trust management in CIDS	No	Focuses on enhancing trust in collaborative intrusion detection
Kumar et al. [37]	Collaborative trust mechanism using vehicular communication	Vehicular communication	No	Innovative in using vehicular communication for trust

Continued on next page

Table 4.1 – continued from previous page

Author(s)	**Main Focus/Contribution**	**Context/ Domain**	**Integration with Blockchain**	**Notable Remarks**
Li et al. (2013) [16]	Intrusion sensitivity for identifying insider attacks	Insider attacks	No	Addresses robustness and trust
Fung et al. (2014) [36]	Use of historical data and machine learning for accurate decision-making	Decision-making in CIDS	No	Leverages machine learning for improved decision-making in CIDS
Meng et al. (2017) [15]	Emphasis on Bayesian inference	Bayesian inference in CIDS	No	Applies Bayesian methods for more accurate intrusion detection
Li et al. (2021) [18]	Hybrid trust management approach	Trust management in CIDS	No	Combines multiple trust management techniques for robustness
Fung et al. (2016) [17]	Use of hypothesis testing	Hypothesis testing in CIDS	No	Innovative use of statistical methods in intrusion detection
Meng et al. (2019) [65]	Blockchain-based trust management technique	Trust management in CIDS	Yes	Addresses communication and computational overhead
Alexopoulos et al. [13]	Exploration of Blockchain technology and CIDS	Blockchain and CIDS	Yes	No concrete implementation proposed

4.2 Argumentation Framework applied to collaborative intrusion detection

Collaborative Intrusion Detection Systems (CIDS) are a subclass of multi-agent systems [66–68] that use a swarm of IDSs to monitor a network for any malicious activity. In this complex network, each agent or IDS acts independently, which might lead to circumstances where the results are inconsistent or lacking [2, 3, 22, 57]. This complexity seriously threatens the reliability and speed with which intrusion detection can alert administrators of attacks on their networks and enable rapid countermeasures. In light of these obstacles, it is crucial to have a novel strategy that encourages collaborative reasoning and decision-making among multiple IDSs.

In these cases, the Argumentation Framework (AF) is helpful. Since the AF effectively controls complex multi-agent systems [69], it can significantly improve the functioning of CIDS. It provides a logical framework for making quick, informed decisions despite conflicting data.

Due to the similarities between CIDS and multi-agent systems and the effectiveness of AF in managing such systems, using AF in the context of CIDS is not only viable. Still, it has the potential to revolutionize their management.

In the following sections, we will look more closely at the intricacies of integrating AF with CIDS, including the argument's structure, building the Argumentation Framework, conflict resolution, decision-making, and a practical illustration.

4.2.1 Argument structure

To establish a trustworthy model for intrusion detection systems, we need a well-defined argument structure that can capture the complexity and dynamism of multiple events and their relationships.

Starting with the Event Calculus framework [70], we have developed an argument structure that connects the complexity of intrusion detection systems with the logical consistency of the Argumentation Framework. Our proposed argument structure combines the Event Calculus's flexibility and the Argumentation Framework's logical precision. Here is an explanation of how it works:

Argument(*IDS, DetectionEvent, Confidence, Premises*)

- **IDS**: "IDS" stands for Intrusion Detection System identifier or name. It helps to provide context for the detection event within the system.

- **DetectionEvent**: This element identifies the event detected by the IDS, such as "Malware," or may indicate "none" if there was no detection. This element serves as the basis for the argument's conclusion. Based on the Event Calculus construct HoldsAt, our argument concludes that if HoldsAt(Detection(e), t) is true according to the given premises, then the event e is considered an intrusion. This convergence of logical constructs enhances the accuracy and context-aware detection of potential intrusions.
- **Confidence** is a number that indicates the probability that an event is an intrusion (in cases of detection events) or the absence of intrusion (in cases of non-detection events). This value ranges from 0 to 1, and it serves as an evaluative indicator of the accuracy of the DetectionEvent.
- **Premises List** is crucial to our argument structure, encompassing systems state s and external events and premises specific to the argumentation framework.
  - Happens(ExtEvent(e_{ext})): This premise serves as the sensory input to the system when an external event (e_{ext}) is detected by the intrusion detection system. These events can vary from minor anomalies to apparent signs of intrusion.
  - Initiated(e, p, t): This premise delves into the system's internal workings. It refers to the start of an event, noted e, activated by a property p of the state s at a specific time t. The state of the IDS, denoted as s, encompasses its current setup, which includes its rules, signatures, policies, actions, and configuration preferences. Event Calculus is a conjunctive logical formula using predicates (Initiates(ExtEvent(e_{ext}), Detection(e), t), Happens(ExtEvent(e_{ext}), t)).

Furthermore, we propose two original premises exclusive to the argumentation framework: ' supports' and 'attacks.' In the previous section, we introduced the argumentation logic and discussed how different arguments are interconnected within the framework. We will delve deeper into these premises in the next section.

To illustrate the argument's structure, we have included listing 4.1 that shows an example based on a real IDS alert taken from the SABU Dataset [71], and we have set the confidence level to 1 in this example.

Listing 4.1 JSON formatted argument

```
{
  "IDS": "cz.casablanca.nemea.
  blacklistfilter",
  "DetectionEvent": {
```

```
      "Event": "Malware",
      "ID": "609cb3f9-95ac-445a-a132-
      aa38023af091"
    },
    "Confidence": 1,
    "Premises": {
      "Happens(ExtEvent)": {
        "EventTime": "2019-03-10T23:55:59Z",
        "CeaseTime": "2019-03-11T00:02:09Z",
        "Proto": ["tcp"],
        "SourceIP": ["64.110.230.41",
        "71.99.142.131"],
        "FlowCount": 334,
        "InFlowCount": 290,
        "OutFlowCount": 44,
        "ByteCount": 1600063,
        "PacketCount": 3031
      },
      "Initiated(e, p, t)": {
        "e": "Malware",
        "p": ["Flow", "Blacklist"],
        "t": "2019-03-10T23:55:59Z",
        "s": {
          "IDS": "cz.casablanca.nemea.
          blacklistfilter",
          "Type": ["Flow", "Blacklist"],
          "AggrWin": "00:05:00"
        }}},...
  }
```


Prolog Implementation of Event Calculus

We aimed to make the theoretical underpinnings of our argument structure more concrete by implementing them within the Event Calculus framework. To achieve this, we utilized Prolog, a declarative programming language known for its logical reasoning and symbolic computing capabilities. The Prolog code 4.2 provided represents the various state compo-

nents of our intrusion detection system, including rules, signatures, policies, and settings. We model real-world intrusion scenarios by defining external events and time points. The code can be extended to accommodate different types of external events and their identification criteria through rules and signatures. The initiates predicate is crucial in connecting external events with their effects on the system's state. This Prolog implementation provides a practical representation of our argument structure and demonstrates how the Event Calculus framework can smoothly integrate with the complexities of intrusion detection.

Listing 4.2 Abstracted Prolog Implementation for IDS using Event Calculus

```
% State components
rules(RuleType).
signatures(SignatureType).
policies(_).
configuration(_).

% State definition
state(s) :- rules(_), signatures(_), policies(_),
    configuration(_).

% External events
external_event(_,_,_).

% Time points
time(T).

% Define a specific pattern
pattern_type(PatternWindow).

% Signatures and rules
rule_definition(Event, T) :-
    findall(_, (happens(Event, _)), EventList),
    length(EventList, EventCount),
    EventCount >= Threshold,
    T>=TimeThreshold.

signature_definition(external_event(_, _, _),Pattern) :-
    pattern_type(Pattern).

% Modify the initiates predicate to check for external events
initiates(external_event(Source,Target,Detail),detection_type,Time) :-
    rule_definition(external_event(Source,Target,Detail), Time);
    (pattern_type(time_window(Window)), Time>Window).

% Termination rule (abstracted for simplicity)
terminates(_,_) :- fail.

% Adding sample external events
happens(external_event(Source,Target,Detail),Time).

% Predicates representing initiation, termination, and holdsAt
initiated(DetectionEvent,RuleType,Time) :-
```

```
    initiates(ExtEvent,DetectionEvent,Time),                                    43
    happens(ExtEvent,Time).                                                     44
                                                                                45
terminated(Event,Time) :-                                                       46
    terminates(Event,Time),                                                     47
    happens(Event,Time).                                                        48
                                                                                49
holdsAt(Event,Time) :-                                                          50
    initiated(Event,RuleType,Time),                                             51
    \+ terminated(Event,Time),                                                  52
    (rules(RuleType);signature(SignatureType)),                                 53
    time(Time).                                                                 54
```

The Prolog code offers a conceptual foundation for an abstracted IDS that utilizes event calculus. The code starts by establishing the components of the IDS, encompassing rules, signatures, policies, and settings (lines 2,3,4,5). Each component is denoted by a predicate that accepts different values.

The state predicate (line 8) represents the comprehensive state of the system, which is determined by a combination of its rules, signatures, policies, and configurations. The code further incorporates a generic *external_event* entity, representing any external event an IDS may monitor.

The time is represented by the time(T) predicate (line 15), which enables the system to monitor and analyze events occurring within a chronological framework effectively. The predicate $pattern\_type(PatternWindow)$ (line 18) denotes the particular patterns or behaviors the IDS intends explicitly to identify.

The fundamental logic of the IDS is supported by the predicates $rule\_definition$ and $signature\_definition$. The predicate $rule\_definition$ (line 21) is designed to evaluate if a certain number of external events above a predetermined threshold occur during a given period. The predicate $signature\_definition$ (line 27) establishes a relationship between external events and specific attack patterns.

The initiates predicate (line 31) assesses whether a detection event has been started by evaluating the specified rules and patterns. The abstraction of event completion is implemented in this model by consistently returning a false value to enhance simplicity, indicating that no event was terminated.

The code also provides a placeholder for adding sample external events with the happens predicate (line 39). The concluding segment of the code encompasses the definition of predicates (initiated, terminated, and holdsAt) that ascertain the condition of an event at a particular time. Using these predicates (lines 42, 47, 50) enables the system to logically reason the current state of a given event, determining whether it is presently active or has

concluded.

The provided Prolog code presents a systematic framework for an IDS, enabling it to identify particular patterns or behaviors in external events through event calculus.

4.2.2 Building the Argumentation Framework

Constructing an AF for an IDS is a systematic process. Aggregating arguments from various IDSs is essential for identifying conflicting or supporting relationships.

Using established relationships, a final decision can be made regarding potential coordinated attacks, such as DDoS or multi-stage attacks. According to Dung's concept [58], the argumentation process involves attack relations between various arguments.

In the context of intrusion detection, it is easier to establish relationships of support between attack relations using correlation indicators. Expanding on Dung's proposal and defining support relations makes it possible to identify attack relations and capture a broader range of potential attacks between arguments.

Support Relation

One argument supports another due to a correlation between the observed and detected events of the different IDSs. The relationship is specified in the argument using the following premise:

$$\textbf{supports}(\textit{supported\_arg},\ CI_1,\ \ldots,\ CI_n)$$

where n is the number of correlation indicators CI that were identified with the supported argument (*supportedArg*).

The Identified types of correlation are:

- Similarity Indicator (SI) [72, 73]: This metric shows the similarity cluster the argument belongs to based on the clustering model, which matches external occurrences that caused a certain DetectionEvent with the baseline arguments. The value can be either numeric or categorical. Clustering arguments have been used before in [69].
- Temporal Support Indicator (TSI) [72, 73]: This indicator denotes a specific duration within which additional arguments may be deemed supportive, specifically about temporal proximity (e.g., +/- acceptedTimeDifference).

- Spatial Support Indicator (SSI) [18, 73]: The indicator, as mentioned above, refers to a collection of spatial parameters such as IP address ranges or subnets, which must be met by other arguments to be deemed supportive in terms of spatial proximity (e.g., [SourceIPRange, TargetIPRange]).
- Attack Patterns Indicator(API) [72, 73]: It denotes a collection of attack techniques that imply a supportive relationship upon detection by an alternate IDS. Identifying similarities in attack patterns can aid in correlating arguments that capture various stages or components of a multi-stage attack. Consider a scenario where an intrusion detection system (IDS) detects port scanning activity and an Attack pattern Indicator predicts a 'buffer overflow' event pattern. If another IDS detects a buffer overflow, a correlation between the arguments is established, suggesting the occurrence of a multi-stage attack. In [74], a proposed framework aims to predict potential future detection events.

Our main objective is establishing the attack relationship between the arguments to build the formal argumentation framework. Therefore, it is crucial to aggregate the previously mentioned correlation indicators to define a single support relation between two arguments. Therefore, we propose the following formalization of the support relation.

Definition 4.2.1 (Support Relation)**.** *We define the support relation between arguments A and B, denoted by $A \subseteq B$, using the indicators as follows:*

$$w(SI) + w(TSI) + w(SSI) + w(API) = 1;$$

where w is a weight function that takes an indicator and assigns a non-negative weight. $A \subseteq B$ if and only if:

$$\begin{aligned} & w(SI) \cdot (SI(A) = SI(B)) \\ & \quad + w(TSI) \cdot (TSI(A) \cap TSI(B)) \\ & \quad + w(SSI) \cdot (SSI(A) \cap SSI(B)) \\ & \quad + w(API) \cdot (API(A) \cap API(B)) \\ & \quad \geq \mathit{Threshold} \end{aligned}$$

Threshold refers to a predetermined value that assesses the degree of support necessary for a given relationship to be valid. The values enclosed within the parentheses are binary, wherein the value 1 denotes the fulfillment of the condition, and the value 0 signifies its non-fulfillment. When the weighted sum of the fulfilled conditions meets the threshold, it can be

inferred that argument A supports argument B, indicating a potential association between the events they describe.

Attack Relation

The fundamental principle of argumentation theory centers on the notion of "attack relations" among arguments. In contrast to supportive relationships reinforcing one another, the notion of attack relationships introduces the potential for conflict and inconsistency among arguments.

The fundamental principle of argumentation theory centers on the notion of 'attack relations' among arguments. In contrast to supportive relationships reinforcing one another, the notion of attack relationships introduces the potential for conflict and inconsistency among arguments. The key idea is the concept of "attack relations" between arguments. The attack relation is specified in the argument using the following predicate:

$$\textbf{attacks}(attacked\_arg)$$

Definition 4.2.2 (Attack Relation)**.** *Let A and B be two arguments included in the set of arguments AR, and let SG_1, SG_2, ... be predefined sets of DetectionEvents that are supposed to belong to the same cluster and therefore have the same SI. These sets result from a clustering model a detection node applies to past observations. We define the attack relation between A and B, denoted by $A \otimes B$, as follows:*

$A \otimes B$ if and only if:

- *$DetectionEvent(B) \in SG_i$ for some i and*
  $SI(B) \neq SI(SG_i)$, and A is a modified instance of B with $SI(A) = SI(SG_i)$
  or
- *$\exists C \in AR$ such that $B \otimes C$ and $A \subseteq C$.*

The first criterion refers to arguments that belong to the same predefined set of similar *DetectionEvent*s, as denoted by the Similarity Indicator (SI).

When argument A has a DetectionEvent that belongs to a specific set SG_i, and its SI does not match the SI of SG_i, it indicates that there is a significant change in the behavior or characteristics of the observation that might suggest an attack. When the SI of a modified instance of A, called argument B, matches the SI of the corresponding set, B is considered to be in an attack relation with A. In the second criterion, argument A is also deemed

to attack argument B if there exists an argument C such that B attacks C ($B \otimes C$) and A supports C ($A \subseteq C$). This condition enables the argumentation framework to describe the interdependencies among various arguments and to detect indirect attacks through these interdependencies.

Intrusion Detection Argumentation Framework

The Argumentation Framework for the Intrusion Detection System is based on the principles of the Dung argumentation theory. It uses attack relationships within its structure to ensure collaborative intrusion detection.

Definition 4.2.3 (IDS Argumentation Framework (IDSAF))**.** *Let $IDSAF = (N, R)$ be an IDS argumentation framework, where:*

- *N is a set of nodes representing arguments, where each node $n \in N$ is defined as a tuple.:*
  $n = Argument(IDS, DetectionEvent, Confidence, Premises)$.
- *R is a set of directed edges denoting attack relations among arguments.*

$$R = \{(n_i, n_j) \in N \times N \mid n_i \otimes n_j\}$$

n_i and n_j denote the i^{th} and j^{th} arguments, respectively, and it is a directed edge, which represents an attack relation, from argument n_i to argument n_j.

4.3 Conclusion

This chapter has presented the concept of formal argumentation frameworks for intrusion detection systems. These frameworks aim to facilitate collaborative intrusion detection by capturing the relationships among arguments and reasoning about their support and attack relations.

The chapter introduced the support relation, which specifies the correlation between arguments based on different indicators such as similarity, temporal support, spatial support, and attack patterns. These indicators help establish the degree of support between arguments, indicating a potential association between the events they describe. The support relation was formalized using a weighted sum of indicators and a threshold for determining the validity of the relationship.

Furthermore, the chapter introduced the attack relation, representing potential conflicts and inconsistencies among arguments. The attack relation captures the notion of arguments attacking each other directly or indirectly through supporting other arguments. The attack relation was formalized based on the similarity indicator and the presence of supporting arguments.

Lastly, the chapter presented the Intrusion Detection System Argumentation Framework (IDSAF), which utilizes attack relations to structure the relationships among arguments. The IDSAF provides a structured and formal way to reason about the collaborative intrusion detection process.

Overall, formal argumentation frameworks offer a valuable approach to enhancing the effectiveness of intrusion detection systems. These frameworks enable more robust and collaborative intrusion detection by capturing the relationships among arguments and reasoning about their support and attack relations. The presented concepts and formalizations provide a strong foundation for future research and development in intrusion detection.

In the next chapter, we will explore the implementation of the argumentation framework in an actual intrusion detection system and evaluate its effectiveness in collaborative intrusion detection.

CHAPTER 5 ARCHITECTURE AND IMPLEMENTATION OF XM-GUARDS

In the previous chapters, we discussed collaborative intrusion detection systems (CIDS) 2, blockchain technology 3, and argumentation frameworks applied to intrusion detection 4. In this chapter, We present our XM-Guards Collaborative Intrusion Detection System that employs an Argumentation Framework deployed on the Blockchain. The used design ensures the development and the sharing of collective knowledge to potentially capture sophisticated attacks (e.g., distributed, simultaneous, and multi-stage attacks) and false/anomalous alarms. In this chapter, we present the details of the architecture and implementation of our system.

5.1 Design and Architecture

The suggested Blockchain-based Collaborative Intrusion Detection argumentation framework was designed to bring together the trustworthiness of blockchain technology with the process of rational decision-making through argumentation. Two critical components of the solution are the guard layer added to the basic intrusion detection system and the Blockchain layer that supports the process behind the argumentation logic.

5.1.1 Blockchain Technology choice

Evaluating the necessity of a permissioned private blockchain for our system involved a comprehensive analysis of several criteria, as indicated by the flowchart in Figure 5.1. The flowchart for blockchain technology choice orientation is a decision-making tool that helps assess whether blockchain is suitable for a use case. It evaluates the need for decentralization, data privacy, intermediaries, trust, scalability, and network participants to recommend whether blockchain is the right choice.

The evaluation of the necessity of a permissioned private blockchain for our system involved a comprehensive analysis of several criteria, as indicated by the flowchart provided in the source.

The need to save the state was apparent due to the requirement of sharing the outcomes of the arguments with other participants. This practice guarantees all participants equal access to the results and can make well-informed decisions based on the information.

Additionally, the system has been specifically developed to support the participation of several writers. This aspect is of utmost importance as it ensures that each participant has the

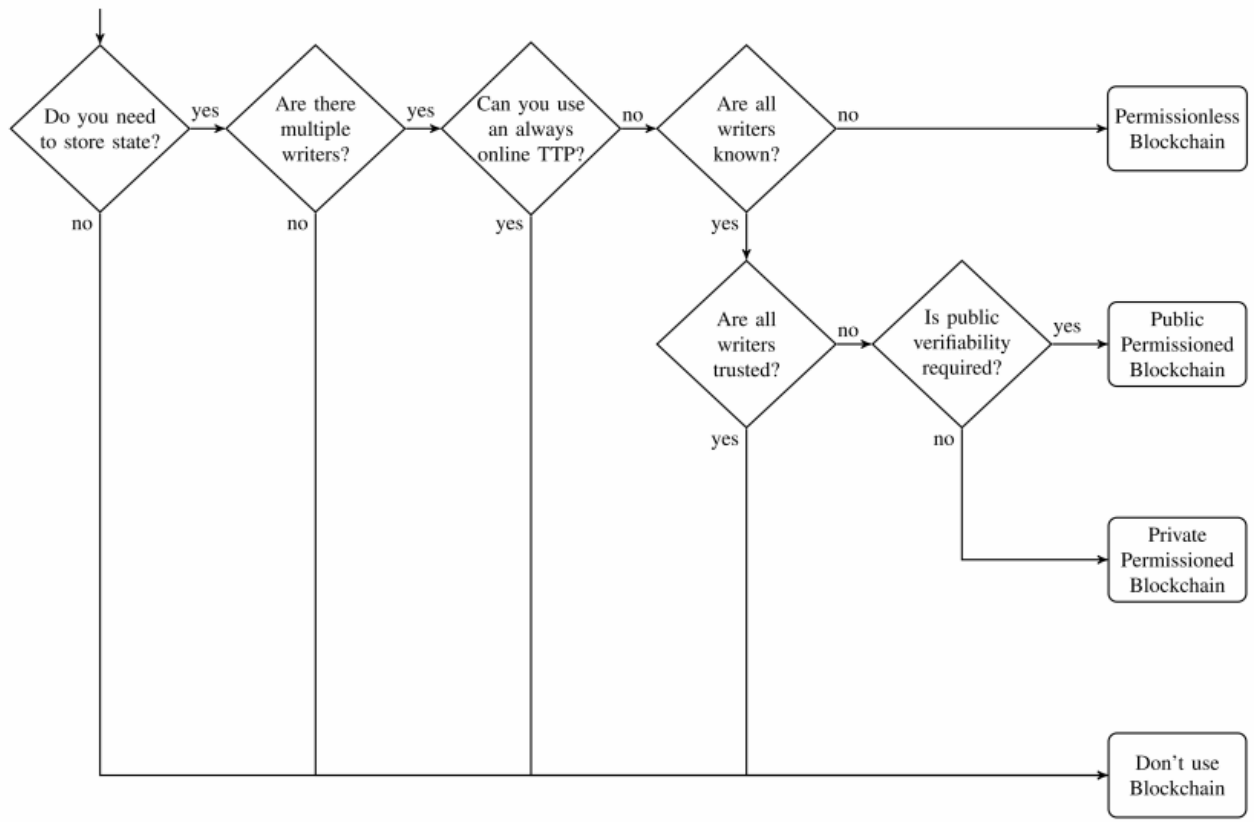

Figure 5.1 Flowchart for blockchain technology choice orientation [5]

potential to launch a discourse. Within the framework of our system, the term "argumentor" may be seen as interchangeable with the term "writer," highlighting the decentralized structure of the platform, whereby each member is afforded an equal opportunity to make contributions.

Upon careful examination of the infrastructure, it became evident that depending on a constantly connected Trusted Third Party (TTP) was not viable. The primary objective is to develop a zero-trust environment, eliminating the need to rely on confidence in any external entity. This strategy effectively improves the security and autonomy of the system.

Moreover, it is essential to note that all participants who engage in the system as authors or players are identifiable entities. This is accomplished by strategically deploying a certain number of nodes who serve as representatives of our proponents. The presence of these entities serves to preserve the integrity of the system by ensuring that only recognized members are permitted to engage in the process of argumentation.

Nevertheless, it is essential to acknowledge that although all writers possess recognition, they do not automatically maintain credibility. Our proposed solution's fundamental concept

is to function inside a trustless setting. Trust is not unconditionally bestowed onto any participating node. Logical argumentation plays a vital role in this context, ensuring that reasoned deliberation leads to conclusions instead of solely relying on trust.

Finally, it should be noted that the system does not need public verifiability. The adequacy of verification within the enterprise-scale context is considered sufficient. The verification methods do not need external or public entities, hence maintaining the privacy and exclusivity of the system for its intended participants.

In conclusion, these considerations underscore the rationale for opting for a permissioned private blockchain tailored to our argumentation system's specific needs and challenges.

Ethereum Clique

In the context of a private and permissioned blockchain, the Clique consensus mechanism in Ethereum, which operates based on Proof-of-Authority (PoA), is a suitable option compared to alternative consensus algorithms. In contrast to the energy-intensive Proof-of-Work (PoW) mechanism, Clique offers a more energy-efficient alternative that eliminates intricate mathematical problem-solving requirements. The faster block timings of this system contribute to initiated transaction confirmations, rendering it highly appropriate for private networks characterized by pre-identified and trusted members. In contrast to the Proof-of-Stake (PoS) consensus mechanism, the Clique consensus mechanism does not need players to stake tokens. This characteristic of Clique contributes to creating a more predictable network. It ensures deterministic block timings. In addition, Clique distinguishes itself from Delegated Proof-of-Stake (DPoS) by mitigating the risks associated with centralization and the possibility of collusion among validators. This is achieved by not depending on a restricted set of chosen validators. Finally, it should be noted that although Byzantine Fault Tolerance (BFT) algorithms can manage rogue nodes, the simplicity of the Clique method renders it more convenient for implementation in private and consortium chains. Clique's combination of security, velocity, and energy preservation makes it an interesting choice, especially for networks prioritizing trustworthiness and efficiency.

To further elucidate the advantages and characteristics of the Clique consensus mechanism in comparison to other consensus algorithms, the following table 5.1 provides a detailed comparison:

Table 5.1 Comparison of Ethereum's Clique with Other Consensus Algorithms

Criteria	**Clique (PoA)**	**Proof-of-Work (PoW)**	**Proof-of-Stake (PoS)**	**Delegated PoS (DPoS)**
Energy Efficiency	High	Low	Moderate	Moderate
Block Time	Fast	Variable	Faster	Fast
Security	High	High	High	Moderate
Centralization	Low	Risky	Risky	High
Implementation Complexity	Moderate	High	High	Moderate
Suitability for Private Networks	High	Low	Moderate	Moderate

5.1.2 Design and components of XM-Guards

Figure 5.2 shows an instance of the global architecture of the XM-Guards framework that scales over four different organization networks, each representing a domain. Four domains are shown, labeled A, B, C, and D, but the framework is designed to accommodate any number of domains. Each domain may have its specific composition of Intrusion Detection Systems (IDSs), including variations in Network-based IDS (NIDS), Host-based IDS (HIDS), and honeypots. These IDSs interact with each other through the Guard layer, ensuring their connection to the argumentation framework deployed on the blockchain. The SABU Dataset, used in our experimentations, is a real-world example of such a network as the alerts were collected from heterogeneous intrusion detection systems deployed across several organizations [71].

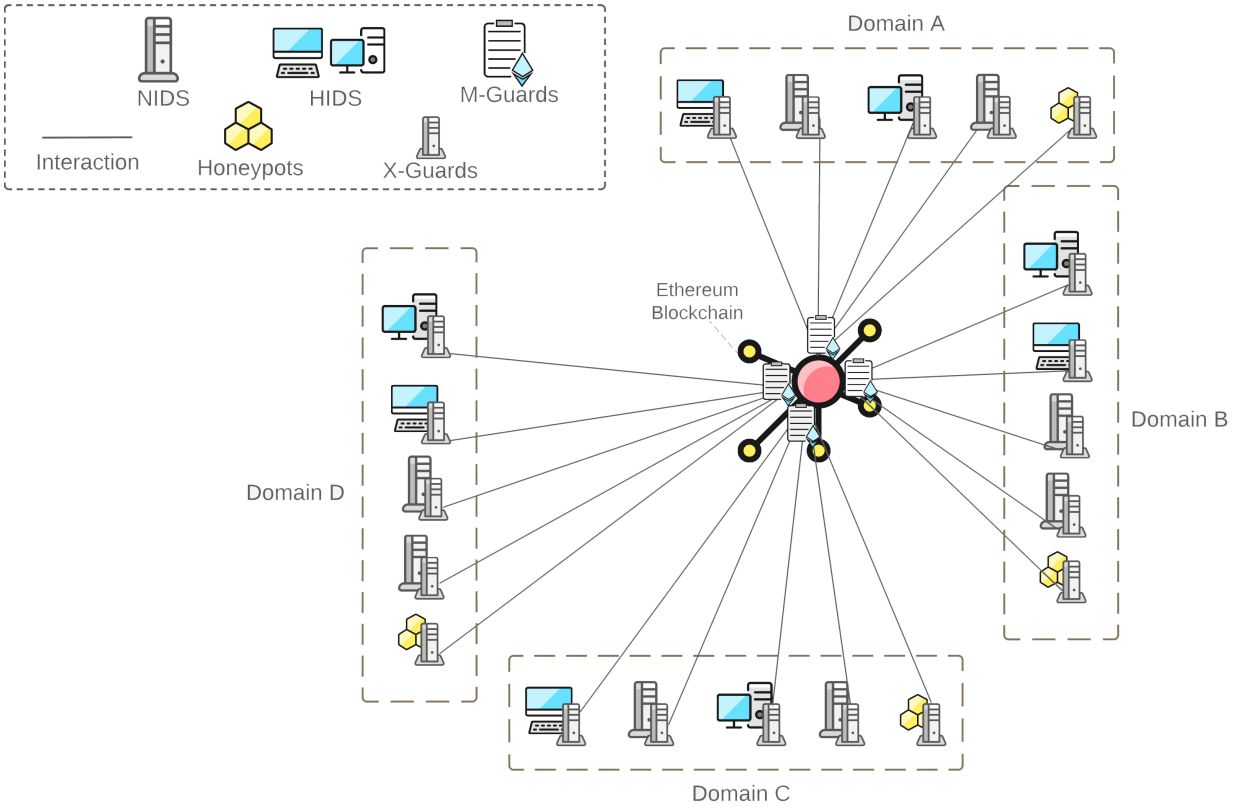

Figure 5.2 Architecture instance of the XM-Guards

Before diving through the architecture of the Guard layer component, we describe the actors involved, shown in Figure 5.3.

- **Organization network/ System activity**: inputs raw data to the X-Guard component. Events on the organization's network or a single host inside the network are

the primary sources of this information. Network traffic patterns, user actions (such as attempting to access restricted files), system logs, and application activity are all examples of possible events.

Since all actions performed on a network generate traffic data, it is possible to get valuable insight into potential security issues by analyzing network traffic. The X-Guard processes this raw data.

- **Blockchain Node**: provides the decentralized ledger, hosting the M-Guards that hold the logic of the argumentation process. It ensures that every transaction and result of the argumentation are recorded and accessible to all nodes in the network.
- **Human Expert**: consults argumentation and support graphs to understand the risks thoroughly. They identify contrasting detections using the attack relationships in the argumentation graph and prioritize serious dangers utilizing the weight of arguments in the support graph. The human expert uses the framework's view of network hazards to get an informed security viewpoint. They can also manually customize the generation of the temporal and spatial indicators associated with the arguments.

After providing an overview of the stakeholders participating in the blockchain-based CID argumentation framework, our focus shifts towards the fundamental components of the system, namely the elements of the guard layer (X-Guards and M-Guards) illustrated in Figure 5.3.

- **Detection Engine**: serves as the primary stage of data processing in conjunction with an Intrusion Detection System (IDS), whether it be a Network-based IDS (NIDS), Host-based IDS (HIDS), or Honeypot. It is responsible for receiving external events, e.g., system activity and network traffic, and detecting possible intrusion events.
- **Correlation Module**: conducts the processing of intrusion events identified by the Detection Engine. The system uses an internal correlation mechanism to eliminate instances of duplication. Furthermore, it produces correlation metrics derived from these occurrences. Those indicators are essential for developing arguments within the argumentation framework.
- **Argumentation Module**: crafts arguments from correlated events and corresponding indicators. The primary function of this entity is to examine the correlation between an external argument and its internal counterparts through the evaluation of relevant indicators. The assessment process can either indicate a 'support' relationship and

transfer the argument to the Communication Module or identify an 'attack' relationship if the observed event matches up, but the detection event differs. When no correlation is present, the module chooses to remain silent and abstain from transferring the argument to the Communication Module.

- **Communication Module**: listens to the Blockchain events, receives arguments that initiate the argumentation process, and accordingly interacts with the Blockchain using its API and the established publish-subscribe system. The necessary 'attack' or 'support' methods from the smart contract in the M-Guard component are called into action due to the associations made in the Argumentation Module. It also plays a crucial role in inter-module and network communication by monitoring the outcomes of the argumentation process.

- **Human Machine Interface (HMI)**: displays the argumentation process outcome, visually representing the attack and support graphs. This allows security analysts to assess the network's current status and take necessary action.

- **Decision-Making Module**: This module is deployed on the blockchain and represents the primary function of the M-Guard Component, defined on a smart contract. The module is illustrated on a smart contract, a computer program automatically managing events and actions. It creates an argumentation graph to define arguments and their relationships. Additionally, it permits X-Guards to either launch an attack or support arguments. Through a series of computations, the module resolves conflicts calculates weights, and prepares the visualization for the X-Guard Component.

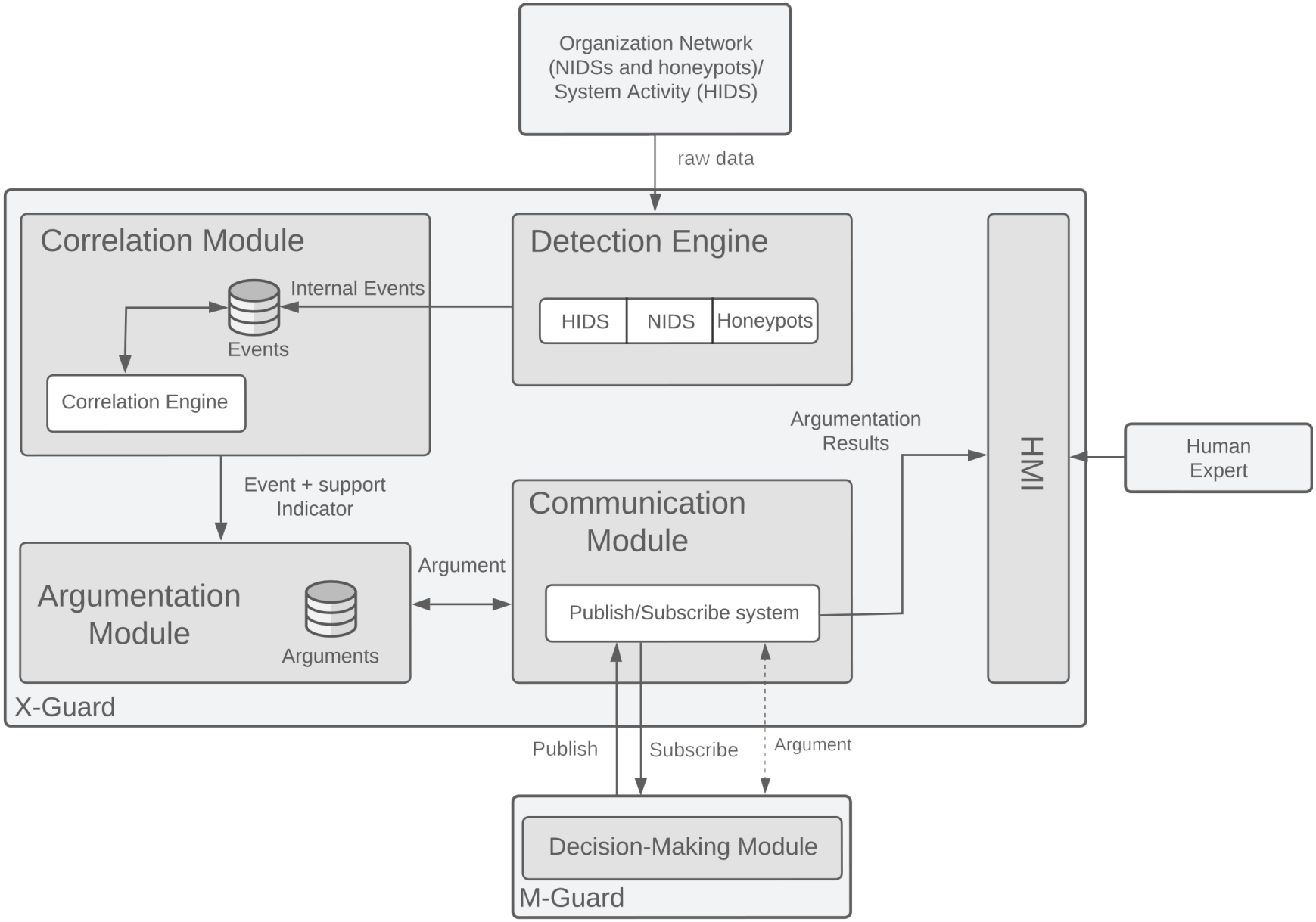

Figure 5.3 Components of the Guard Layer (M-Guards and X-Guards)

5.2 Implementation

The design of the Blockchain-based argumentation framework requires the critical stage of system implementation. The implementation is divided into on-chain and off-chain components. The on-chain implementation involves executing components (M-Guards) within the Blockchain network, mainly through smart contracts. The off-chain implementation includes components external to the Blockchain network (X-Guards). The subsequent sections provide an in-depth analysis of these aspects, clarifying their roles, functions, and operational mechanisms within the argumentation framework. Effective collaborative intrusion detection across multiple organization networks is achieved by harmonizing on-chain and off-chain implementations.

5.2.1 On-chain implementation

We use a permissioned Distributed Ledger Technology (DLT) that supports smart contracts for our on-chain deployment. This component relies on a permissioned Ethereum blockchain running on a distributed network of nodes. We use the Proof-of-Authority (PoA) consensus method [50] to provide a faster, more efficient alternative to conventional Ethereum networks' energy-intensive Proof-of-Work consensus mechanism [45,47]. Proof of Authority (PoA), also introduced as clique [50], does not rely on solving complex mathematical problems. Instead, most "authorities" signatures are necessary to issue a new block. These authorities are nodes granted the privilege of adding new blocks and ensuring the Blockchain's integrity.

This on-chain setup guarantees the efficiency, transparency, and traceability of Blockchain technology and offers a solid foundation for implementing our reasoning logic and procedures. The on-chain implementation is well-suited for the cooperation of many organizational networks in collective intrusion detection since it closely follows the structure and requirements of our argumentation framework. We implemented the argumentation framework (AF) in a smart contract called "Argumentation."

- The argumentation graph is created and managed using a data structure within the smart contract that defines the arguments as nodes and attack relations as edges. Each X-Guard can initiate an argumentation by providing his argument (*initiateArgumentation(Argument)*). It can also attack (*attack(Argument)*) or support (*support(Argument, Indicator)*) another argument.
- The interaction between X-Guards is automatically handled by employing the event system embedded in the smart contract. The created *ArgumentationInitiatedEvent*

event notifies the X-Guards listening to it about the initiation of the argumentation process.

- Attack relations are automatically computed from the support relations defined in section 4.2.2 when the *getArgumentationGraph* method is called.
- From the obtained AF, an extension can be computed using the method *computeExtensions*. It possibly provides a set of arguments that can lead to a final decision about intrusion detection.

5.2.2 Off-chain implementation

Thanks to the integrated correlation models for clustering and prediction of alerts, each X-Guard can generate indicators such as similarity and attack pattern indicators to participate in the argumentation process thanks to its correlation module. These results are then passed to the argumentation module to construct the argument (*prepareArgument*). An X-Guard can react accordingly when argumentation is initiated by continuously listening to events on the Blockchain (*handleArgEvent*). When receiving an argument, its relation with the latter is automatically computed using the attack and support definition; see section 4.2.2. At any stage of the argumentation process, an X-Guard can request the state of the argumentation to visualize it in the HMI by calling the *getArgumentationGraph*.

Figure 5.5 showcases all possible interactions between an X-Guard, an M-Guard, and other X-Guards (X-Guards Network in Figure 5.5). The process of argumentation is represented as follows. Firstly, an X-Guard prepares an argument with an alert detection. Next, the prepared argument is sent to the M-Guard to initiate an argumentation. Upon receiving the argument, the M-Guard sends an argumentationInitiatedEvent to the X-Guard network. In response, an X-Guard handles the argument, comparing the similarity indicators received and processed locally and considering the detection event. If the similarity indicators match, but the detection events differ the X-Guard attacks that argument. However, if the support threshold is met, the X-Guard supports that argument. Any X-Guard can access the argumentation graph and compute the extensions at any point.

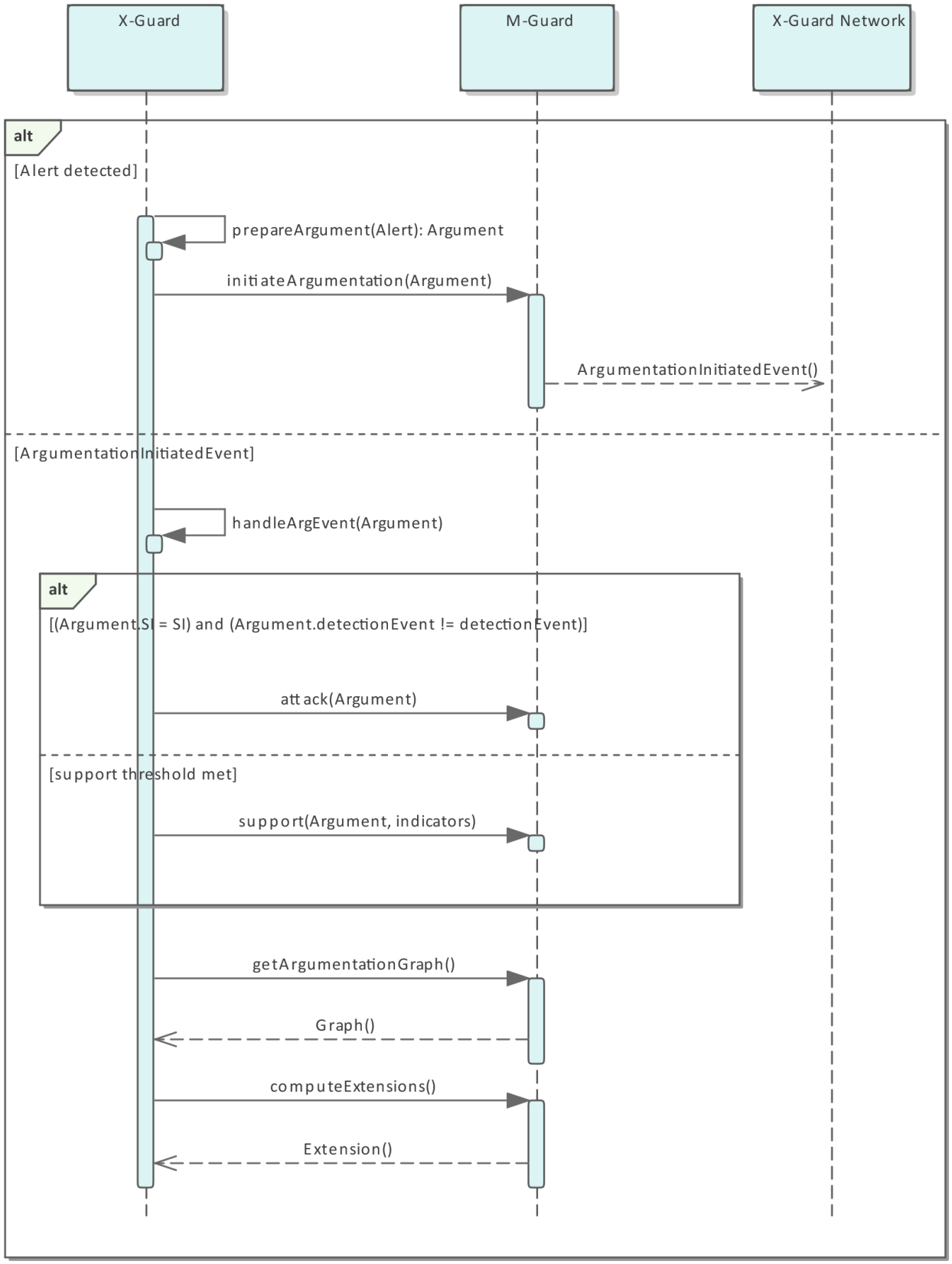

Figure 5.4 Interaction between the XM-Guards different components - Sequential Diagram

5.3 Potential multi-stage attack detection

In the dynamic realm of cybersecurity, the capacity to anticipate and identify prospective threats is paramount. Conventional Intrusion Detection Systems (IDS) have played a crucial role in detecting ongoing threats by analyzing recognized patterns. Nonetheless, in light of the escalating complexity of cyberattacks, it is imperative to detect and avert potential risks before their actualization proactively. The AIDA framework and the Attack Patterns Indicator (API) are introduced here. The API, which represents an innovative methodology, incorporates a sequence of attack strategies that, upon detection, provide insights into possible orchestrated or multi-stage threats. In contrast, the AIDA framework [74], which stands for Analytical Intrusion Detection Alert System, is a cutting-edge mechanism specifically engineered to handle intrusion detection alerts by placing notable importance on predictive analytics and alert correlation. The following section will delve deeper into how these two components intertwine.

5.3.1 AIDA Framework

This framework primarily centers on alert correlation and predictive analytics, aiming to provide a more proactive approach to threat detection and mitigation.

Framework Composition and Functionality

AIDA is an intricately engineered system comprising numerous components, each of which fulfills a distinct function; its primary function is stream processing. The fundamental functionalities of these components consist of:

- **Alert Filtering:** This procedure guarantees the exclusion of potential false positives by considering solely pertinent and genuine alerts.
- **Alert Aggregation:** This element detects and annotates alerts that have the potential to be aggregated, thereby offering a consolidated perspective of comparable threats.
- **Rule Generation:** Through the consolidated alerts, the mining component constructs sequential rules that effectively identify patterns and possible threats. Some examples of mined rules are illustrated in table 5.2. Each rule is composed of an antecedent and a consequent. If the detection event specified in the antecedent happens, it is predicted to be followed by the event set in the consequent column.

- **Predictive Analysis:** The matching component performs a cross-reference between incoming alerts and previously generated rules. Indicating a potential future threat, a predicted alert is generated if a match is detected.

Underlying Infrastructure and Data Flow

The core of AIDA's functioning is centered upon the Apache Kafka message broker, which enables smooth communication across the many components of the framework. Alerts inside the framework undergo processing by the IDEA [75] format, assuring consistency and compatibility.

A systematic and simplified process characterizes the data flow inside the AIDA system.

The initial step in processing security events, structured according to the IDEA format, is ingesting these events by the sanitization component operating on the input subject. Once the warnings have undergone thorough syntactic and semantic evaluations, they are then forwarded to the designated subject in a sanitized manner. The aggregation component subsequently analyzes these warnings, identifying and categorizing those considered aggregates. The items above are subsequently transmitted to the consolidated subject matter. Both the mining and matching components extract information from the aggregated topic. While the former process creates rules, the latter process verifies the existence of these rules in the incoming alert stream. If a match is identified, a forecasted notification is sent to the prediction topic within the Kafka messaging system.

Figure 5.6 showcases the infrastructure and data flow of the AIDA framework. It utilizes the Apache Kafka message broker [76] as the underlying messaging system for communication between its components. Kafka facilitates the distribution of alerts through multiple topics. The first component, the sanitization component, consumes incoming data from an input topic. It performs checks on the syntax and semantics of the alerts and sends the alerts that pass the checks to a sanitized topic. The aggregation component then consumes the sanitized topic and identifies alerts that can be considered aggregates. These marked alerts are returned to Kafka and sent to an aggregated topic. The mining component consumes the aggregated topic and generates rules based on the alerts stored in a database. The matching component also consumes the aggregated topic and searches the stream of incoming alerts for the presence of previously generated rules retrieved from the database. If a rule is found in the stream, a predicted alert is created and sent to the Kafka topic "predictions." Lastly, the feedback component consumes various Kafka topics to measure the performance metrics of the components.

Table 5.2 AIDA Framework: Rule Antecedent to Consequent Mapping Example [1]

Rule	**Antecedent**	**Consequent**
1	cz.cesnet.tarpit_Recon.Scanning_8000	cz.cesnet.tarpit_Recon.Scanning_88
2	cz.cesnet.nemea.hoststats_Recon.Scanning _None	cz.cesnet.hoststats_Recon.Scanning _None
3	cz.cesnet.tarpit_Recon.Scanning _2323	cz.cesnet.tarpit_Recon.Scanning_23
4	cz.cesnet.tarpit_Recon.Scanning _8000	cz.cesnet.tarpit_Recon.Scanning _8080
5	cz.cesnet.tarpit_Recon.Scanning_80, cz.cesnet.tarpit_Recon.Scanning_8000	cz.cesnet.tarpit_Recon.Scanning_8080
6	cz.cesnet.tarpit_Recon.Scanning_88, cz.cesnet.tarpit_Recon.Scanning_8000	cz.cesnet.tarpit_Recon.Scanning_8080
7	cz.cesnet.nemea.hoststats_Recon.Scanning_None, cz.casablanca.nemea.hoststats _Recon.Scanning_None	cz.cesnet.hoststats_ Recon.Scanning_None
8	cz.cesnet.tarpit_Recon.Scanning_8080, cz.cesnet.tarpit_Recon.Scanning _8000	cz.cesnet.tarpit_Recon.Scanning_88
9	cz.cesnet.tarpit_Recon.Scanning_8080, cz.cesnet.tarpit_Recon.Scanning _8000	cz.cesnet.tarpit_Recon.Scanning_80
10	cz.cesnet.tarpit_Recon.Scanning_88, cz.cesnet.tarpit_Recon.Scanning_80	cz.cesnet.tarpit_Recon.Scanning_8080
11	cz.casablanca.nemea.bruteforce_Attempt.Login_23	cz.cesnet.tarpit_Recon.Scanning_23
12	cz.cesnet.tarpit_Recon.Scanning_88, cz.cesnet.tarpit_Recon.Scanning_8080	cz.cesnet.tarpit_Recon.Scanning_80
13	cz.cesnet.tarpit_Recon.Scanning_2222	cz.cesnet.tarpit_Recon.Scanning_22
14	cz.casablanca.nemea.hoststats_Recon.Scanning_None, cz.cesnet.hoststats_Recon.Scanning_None	cz.cesnet.nemea.hoststats _Recon.Scanning _None
15	cz.cesnet.tarpit_Recon.Scanning_8080, cz.cesnet.tarpit_Recon.Scanning_81	cz.cesnet.tarpit_Recon.Scanning_80
16	cz.cesnet.tarpit_Recon.Scanning_80, cz.cesnet.tarpit_Recon.Scanning_81	cz.cesnet.tarpit_Recon.Scanning_8080
17	cz.cesnet.hugo.haas_dionaea_Recon.Scanning_445	cz.cesnet.tarpit_Recon.Scanning_445
18	cz.cesnet.hoststats_Recon.Scanning_None, cz.cesnet.tarpit_Recon.Scanning _2323	cz.cesnet.tarpit_Recon.Scanning_23
19	cz.casablanca.nemea.bruteforce_Attempt.Login_22	cz.cesnet.nemea.bruteforce _Attempt.Login_22
20	cz.cesnet.hugo.haas_cowrie_Attempt.Login_22	cz.cesnet.nemea.bruteforce _Attempt.Login_22

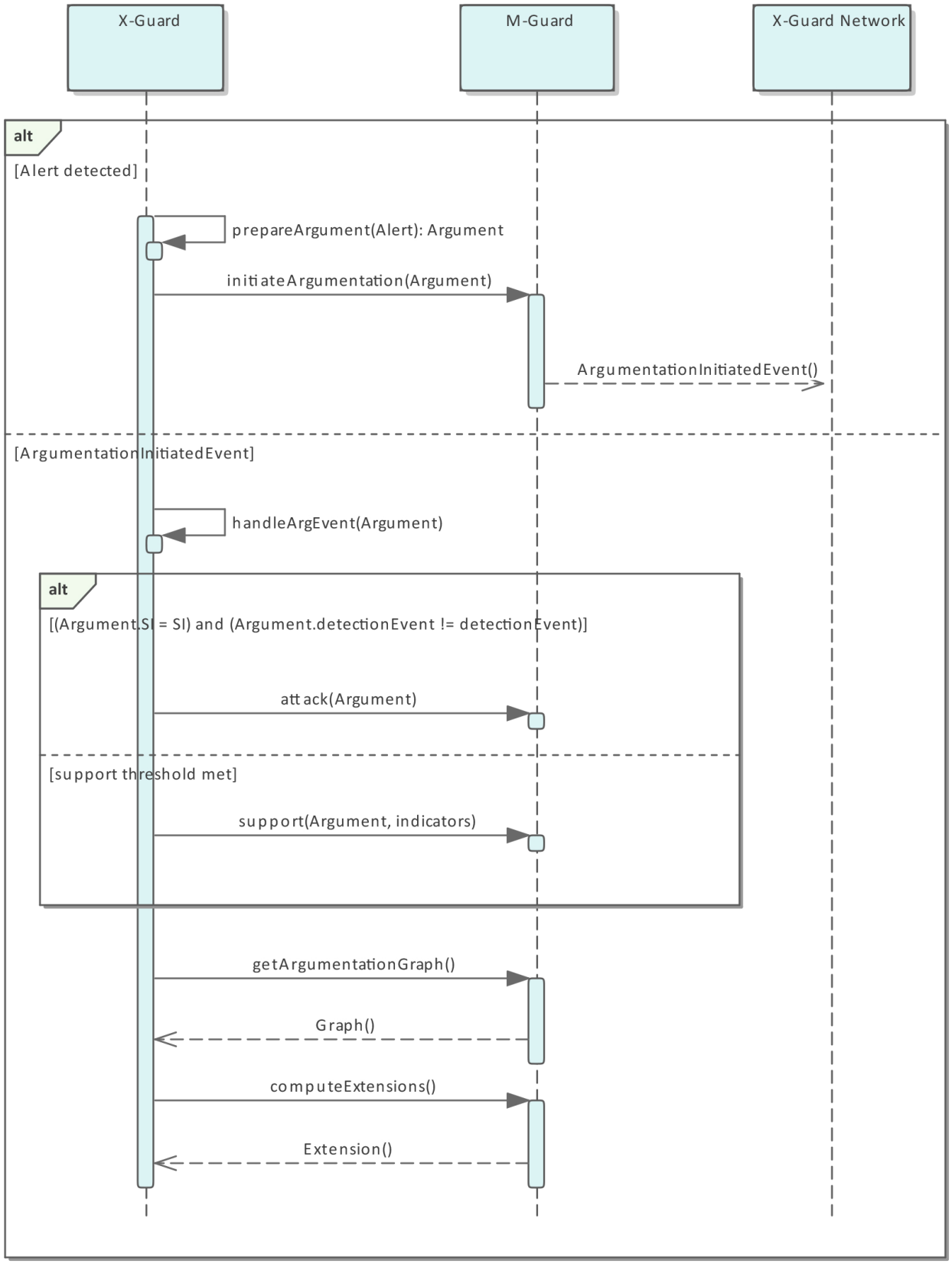

Figure 5.5 Interaction between the XM-Guards different components - Sequential Diagram

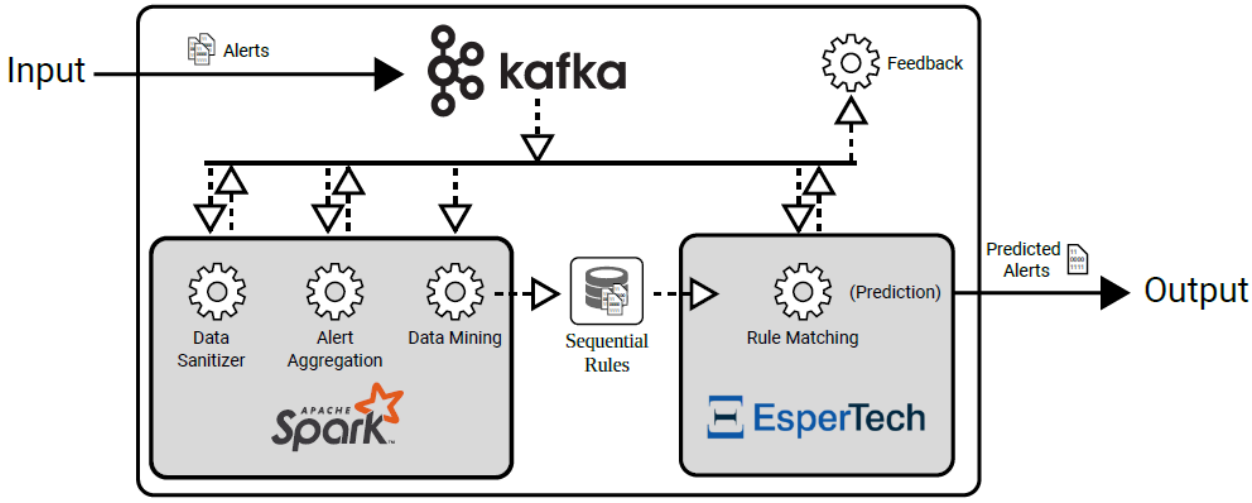

Figure 5.6 AIDA framework architecture

5.3.2 Generating Attack Pattern Indicator using AIDA Framework

Attack Patterns Indicator (API) is a collection of attack techniques that, when identified by an intrusion detection system (IDS), indicate a supportive or corroborative relationship between different stages of an attack. The essence of API lies in its ability to capture and correlate various components of a multi-stage attack, thereby enhancing the predictive capabilities of an IDS.

The actual value of API emerges when it aids in correlating disparate alerts or arguments that might indicate different stages of a complex attack. For instance, consider a scenario where one IDS detects a port scanning activity, and an associated API predicts a subsequent 'buffer overflow' event pattern. A correlation is established if a different IDS subsequently detects this buffer overflow. This correlation suggests that the detected activities are not isolated events but components of a coordinated multi-stage attack.

The initial stage of the process API generation process for the argumentation framework is Alert Generation, during which the AIDA framework analyzes incoming security events and detects and associates alerts using predetermined rules and patterns. Subsequently, the Predictive Analysis phase assumes control. During this stage, the predictive component of AIDA examines the interconnected warnings and generates predictions for probable security incidents that may occur in the future. During the transition to the API Argument Formation step, each created argument effectively integrates the anticipated alert provided by AIDA. Incorporating this element is of utmost importance since the expected warning serves as a

manifestation of the Attack Patterns Indicator, providing a glance into prospective sequences of attacks. The technique has a dynamic nature, hence assuring its continued relevance. During the Dynamic Update phase, AIDA incorporates new alerts and generates updated forecasts, resulting in dynamic adjustments to the API arguments. This ensures that the API arguments accurately reflect the most recent threat landscape.

5.4 Conclusion

This chapter presented a comprehensive overview of the XM-Guards Collaborative Intrusion Detection System, a novel approach that integrates an Argumentation Framework with Blockchain technology. The system is designed to enhance the detection and analysis of sophisticated cyber threats by leveraging the collective knowledge and capabilities of multiple organizational networks.

The architecture of XM-Guards is built upon two primary components: the Guard layer, which interfaces with traditional intrusion detection systems, and the Blockchain layer, which underpins the argumentation logic. The choice of a permissioned private blockchain, specifically Ethereum's Clique consensus mechanism, was justified based on several criteria, including the need for decentralization, trustworthiness, and efficiency in a zero-trust environment.

The detailed design of the XM-Guards system was elucidated, highlighting the roles and functionalities of its various components, such as the Detection Engine, Correlation Module, Argumentation Module, Communication Module, Human Machine Interface, and Decision-Making Module. These components work in tandem to process network events, generate and evaluate arguments, and visually represent the outcomes of the argumentation process for human analysts.

The implementation of XM-Guards was described, focusing on the on-chain and off-chain components. The on-chain implementation, executed within a permissioned Ethereum blockchain, utilizes smart contracts to manage the argumentation framework, ensuring transparency, traceability, and efficiency. The off-chain implementation involves external components that interact with the blockchain and contribute to the collaborative intrusion detection process.

Our study also explored the potential of incorporating the AIDA framework into our solution for generating Attack Pattern Indicators (API). The AIDA framework's emphasis on alert correlation and predictive analytics aligns closely with the fundamental objectives of API, which involve identifying and correlating various stages of a multi-stage attack. Integrating the AIDA framework into our solution promises to enhance our capabilities in detecting and mitigating complex cyber threats more robustly and proactively.

CHAPTER 6 EXPERIMENTATION AND RESULTS

This section presents the details of the experimentation setup and the results obtained from implementing our proposed XM-Guards framework. The experimentation was conducted in two parts: on-chain and off-chain.

6.1 On-chain Experimentation

The experimentation involved using a private Ethereum blockchain, specifically the Clique Ethereum private blockchain [50], for the on-chain component. The clique offers efficiency through faster transaction validation, scalability to handle a large volume of data, and cost-effectiveness by avoiding the computational intensity of other consensus mechanisms like Proof-of-Work. Five nodes were deployed using Docker [77], a platform that facilitates the automated deployment, scaling, and management of applications. Docker offers a lightweight, isolated environment that promotes individual node operations while enabling inter-node communication.

6.2 Off-chain Experimentation

In the off-chain component of the experimentation, Web3 [78] was employed in conjunction with Python [79]. Web3 is a library compilation that facilitates user interaction and a local or remote Ethereum node through various communication protocols such as HTTP, IPC, or WebSocket [80]. Web3 connected our off-chain components, specifically the X-Guards associated with IDS systems, and the on-chain components, specifically the M-Guards, residing on the Ethereum blockchain. It also allows the visualization of argumentation results received from the on-chain side in a graph format using the library graphviz [81].

Our clustering model was also developed as a component of the off-chain experimentation process. The utilization of the clustering model plays a pivotal role in generating the similarity indicator. The latter will subsequently be employed in the process of argumentation.

Table 6.1 presents the technical specifications used in our experimentation.

6.3 Description of the used dataset: SABU Dataset

In this section, we present the SABU Dataset, its source and its content.

Table 6.1 Technical Specifications

Component	Tool	Details
On-chain	Clique, Ethereum [50]	Private Ethereum blockchain (Clique), five nodes deployed using Docker. Used to host the argumentation and decision-making process.
Off-chain	Web3, Graphviz, Python [78, 79, 81]	Used to connect off-chain components with on-chain components
Clustering Model	scikit-learn, Python [79]	Used for generating similarity indicators for the argumentation process

6.3.1 Source

The SABU dataset [71] is derived from the SABU sharing platform administered by the Computer Security Incident Response Team (CSIRT) in the Czech Republic. The establishment of this platform is the result of a joint endeavor between the association of universities in the Czech Republic, the Czech Academy of Sciences (CESNET), and the Cybersecurity Team of Masaryk University (CSIRT-MU). The dataset comprises three discrete computer networks inside the Czech Republic: the National Research and Education Network (NREN), a campus network, and a commercial Internet Service Provider (ISP) network.

- **National Research and Education Network (NREN):** This is a backbone network that spans the entirety of the Czech Republic in terms of geography. Network Intrusion Detection Systems (NIDS), including Network Measurement Analysis (NEMEA) [82], Suricata [83], and Flow-Based Traffic Analysis System (FTAS) Intrusion Prevention System (IPS) [84], are used. Furthermore, honeypots and tarpits such as cowrie, dionaea, and LaBrea are used [85, 86].

- **Campus Network:** This network, situated in a singular geographic area, is linked to the NREN. It is noteworthy that NREN IDSs can monitor the campus network traffic. The campus network employs NIDS such as Flowmon ADS [87] and NEMEA and honeypots, including Heralding and honeyscan [88], an in-house flow-based honeypot monitoring system.

- **Commercial Internet Service Provider (ISP):** Geographically isolated, this network is not near the other two networks. It employs NEMEA primarily as its NIDS.

6.3.2 Content

The SABU dataset comprises alerts gathered from the SABU alert-sharing platform during one week, from March 11th to March 17th, 2019. The IDEA format, a modern JSON extension of the IDMEF, stores these alerts. With the three organizations above, 34 intrusion detection systems, honeypots, and additional data sources generated nearly 12 million alerts during this period.

In the dataset, alerts are classified into various categories, which we attempted to define despite the lack of clarity in the original work. These categories encompass a broad range of alerts, including but not limited to:

- Recon.Scanning - This category refers to activities conducted by potential attackers to gather information about a target network or system. It typically involves scanning for open ports, identifying vulnerabilities, and mapping network architecture. In 6.18, an example of a Recon.Scanning alert.

Listing 6.1 Example of Recon.Scanning Alert


```
{
  "_id": {
    "$oid": "644ddb9abb734cf88049405c"
  },
  "DetectTime": "2019-03-11T00:04:43.312146+02:00",
  "Node": [
    {
      "AggrWin": "00:05:00",
      "SW": [
        "Dionaea"
      ],
      "Type": [
        "Connection",
        "Protocol",
        "Honeypot"
      ],
      "Name": "cz.cesnet.hugo.haas_dionaea"
    }
  ],
  "Target": [
```

```
      {
        "Anonymised": true,
        "Port": [
          445
        ],
        "IP4": [
          "192.0.0.0"
        ],
        "Proto": [
          "tcp"
        ]
      }
    ],
    "ConnCount": 2,
    "Format": "IDEA0",
    "Category": [
      "Recon.Scanning"
    ],
    "Source": [
      {
        "Port": [
          3508
        ],
        "IP4": [
          "22.198.228.92"
        ],
        "Proto": [
          "tcp"
        ]
      }
    ],
    "WinStartTime": "2019-03-11T00:00:01.222836+02:00",
    "WinEndTime": "2019-03-11T00:05:01.222721+02:00",
    "ID": "f62537c2-77b8-49c7-a0a2-24c4b81b20f8"
  }
```

For the following alert categories, we will only show essential fields of the alert that vary from one category to another.

- Attempt.Login - This category indicates unauthorized attempts to access a system or network by guessing or exploiting weak login credentials. Attackers may use various techniques, such as brute force attacks or password guessing, to gain unauthorized access.

Listing 6.2 Example of Attempt.Login Alert

```
{
  ...,
  "Node": [
    {
      "Type": [
        "Relay"
      ],
      "Name": "cz.muni.ics.csirt.flowmon_ads"
    }
  ],
  "Target": [
    {
      "Port": [
        23
      ],
      "IP4": [
        "142.252.92.193"
      ],
      "Proto": [
        "tcp",
        "telnet"
      ]
    }
  ],
  "ConnCount": 55
  "Category": [
    "Attempt.Login"
  ],
```

```
    "Source": [
      ...
    ],
    "ByteCount": 179384,
  }
```

- Attempt.Exploit - This category represents attempts to exploit vulnerabilities in software or systems to gain unauthorized access, escalate privileges, or perform malicious actions. Exploits often target specific vulnerabilities in software or configurations.

Listing 6.3 Example of Attempt.Exploit Alert

```
  {
  {
    "Node": [
      {
        "SW": [
          "TippingPoint_NX_NGIPS"
        ],
        "Type": [
          "Datagram",
          "Content",
          "Protocol",
          "Signature",
          "Policy",
          "Heuristic"
        ],
        "Name": "cz.cesnet.ids_collector.tippingpoint"
      }
    ],
    "Target": [
      {
        "Proto": [
          "udp",
          "domain"
        ]
      }
    ],
```

```
    "Category": [
      "Attempt.Exploit"
    ],
    "Source": [
      { ...,
        "Proto": [
          "udp",
          "domain"
        ]
      }
    ],
  }
  }
```

- Malware - This category refers to malicious software's presence or potential presence, such as viruses, worms, or Trojans. Malware can cause harm to systems by compromising confidentiality, integrity, and availability.

Listing 6.4 Example of Malware Alert

```
  {
    "Node": [
      {
        "AggrWin": "00:05:00",
        "SW": [
          "Nemea",
          "blacklistfilter"
        ],
        "Type": [
          "Flow",
          "Blacklist"
        ],
        "Name": "cz.casablanca.nemea.blacklistfilter"
      }
    ],
    "Category": [
      "Malware"
    ],
```

```
    "CreateTime": "2019-03-11T00:02:55Z",
    "Source": [
      {
        "InFlowCount": 290,
        "Proto": [
          "tcp"
        ],
        "InPacketsCount": 2452,
        "OutPacketsCount": 579,
        "OutFlowCount": 44,
        "InByteCount": 1600063,
        "OutByteCount": 97115,
        "IP4": [
          "64.110.230.41"
        ],
        "Port": [
          443
        ]
      },
      {
        "IP4": [
          "71.99.142.131"
        ],
        "Proto": [
          "tcp"
        ]
      }
    ],
    "ByteCount": 1697178,
    "FlowCount": 334,
    "PacketCount": 3031,
  }
```


- Information.UnauthorizedAccess - This category indicates unauthorized access to sensitive or confidential information. It may involve unauthorized users accessing databases, files, or systems containing sensitive data.

Listing 6.5 Example of Information.UnauthorizedAccess Alert


```
{
  "Node": [
    {
      "SW": [
        "Cowrie"
      ],
      "Type": [
        "Honeypot",
        "Connection",
        "Auth"
      ],
      "Name": "cz.cesnet.hugo.haas_cowrie"
    }
  ],
  "Target": [
    {
      "Proto": [
        "tcp",
        "ssh"
      ]
    }
  ],
  "Category": [
    "Information.UnauthorizedAccess"
  ]
}
```


- Availability.DoS - This category relates to Denial of Service (DoS) attacks that aim to disrupt or interrupt the normal functioning of a system or network. Attackers overload the target with a high volume of traffic or resource-intensive requests, rendering it unavailable to legitimate users.

Listing 6.6 Example of Availability.DoS Alert


```
{
  "Node": [
    {
```

```
        "SW": [
          "Suricata"
        ],
        "Type": [
          "Datagram",
          "Content",
          "Protocol",
          "Signature"
        ],
        "Name": "cz.cesnet.ids_collector.suricata"
      }
    ],
    "Target": [
      {
        "Proto": [
          "udp"
        ]
      }
    ],
    "Category": [
      "Availability.DoS"
    ],
    "Source": [
      {
        "Proto": [
          "udp"
        ]
      }
    ]
  }
```


- Anomaly.Traffic - This category identifies abnormal or suspicious network traffic patterns that deviate from the expected behavior. It may indicate potential malicious activity, such as network scanning, data exfiltration, or command-and-control communication.

Listing 6.7 Example of Anomaly.Traffic Alert


```
{
  "Node": [
    {
      "Type": [
        "Relay"
      ],
      "Name": "cz.cesnet.ftas"
    },
    {
      "SW": [
        "FTAS"
      ],
      "Name": "cz.cesnet.gc15",
      "Tags": [
        "Flow"
      ]
    }
  ],
  "Target": [
    {
      "PortCount": 1,
      "ProtoCount": 1,
      "Note": "Counts are measured up to 32 distinct
         values; arrays are cropped to 4 distinct values
         .",
      "InterfaceCount": 1,
      "Interface": [
        "156 (HundredGigE0/0/0/0, R136, 2001:718:0:600:0
          :135:136:10,195.113.235.99)"
      ],
      "IP4Count": 32,
    }
  ],
  "Category": [
    "Anomaly.Traffic"
```

```
    ],
    "Source": [
      {
        "PortCount": 32,
        "ProtoCount": 1,
        "Note": "Counts are measured up to 32 distinct
           values; arrays are cropped to 4 distinct values
           .",
        "InterfaceCount": 1,
        "IP4": [
          "255.171.239.105"
        ],
        "Interface": [
          "144 (Bundle-Ether111, Telecom Italia Sparkle [
             main], 2001:41a8:500:2:0:0:0:42,195.22.215.22
             9)"
        ]
      }
    ],
    "ByteCount": 36722400,
    "FlowCount": 17655,
    "PacketCount": 706200,
  }
```


- Malware.Ransomware - This category explicitly denotes the presence or potential presence of ransomware, malicious software that encrypts files or systems and demands a ransom for their release.

Listing 6.8 Example of Malware.Ransomware Alert


```
  {
    "Node": [
      {
        "SW": [
          "Nemea",
          "blacklistfilter"
        ],
        "Type": [
```

```
            "Flow",
            "Blacklist"
          ],
          "Name": "cz.casablanca.nemea.blacklistfilter"
        }
      ],
      "Category": [
        "Malware.Ransomware"
      ],
      "Source": [
        {
          "InFlowCount": 2,
          "Proto": [
            "tcp"
          ],
          "OutPacketsCount": 13,
          "OutFlowCount": 2,
          "OutByteCount": 1090,
          "InByteCount": 1033,
          "InPacketsCount": 14,
        }
      ],
      "ByteCount": 2123,
      "FlowCount": 4,
      "PacketCount": 27,
    }
```


- Availability.DDoS - This category signifies Distributed Denial of Service (DDoS) attacks that simultaneously target a system or network by overwhelming traffic from multiple sources. DDoS attacks aim to disrupt service availability and make systems or networks inaccessible to legitimate users.

Listing 6.9 Example of Availability.DDoS Alert


```
{
  "Node": [
    {
      "SW": [
```

```
          "Nemea",
          "amplification_detection"
        ],
        "Type": [
          "Flow",
          "Statistical"
        ],
        "Name": "cz.casablanca.nemea.amplificationdetector
           "
      }
    ],
    "Target": [
      {
        "InPacketCount": 411,
        "InFlowCount": 137,
        "InByteCount": 563481
      }
    ],
    "Category": [
      "Availability.DDoS"
    ],
    "Source": [
      {
        "InFlowCount": 139,

        "InPacketCount": 139,
        "OutFlowCount": 137,
        "InByteCount": 8896,
        "OutByteCount": 563481,

        "Type": [
          "Backscatter"
        ],
        "OutPacketCount": 411
      }
    ],
```

```
    "ByteCount": 563481,
    "FlowCount": 137,
    "PacketCount": 411
  }
```


- Malware.Spyware - This category signifies the presence or potential presence of spyware, malicious software designed to gather sensitive information from a device, such as keystrokes, browsing habits, or login credentials. Spyware operates covertly without the user's knowledge or consent.

Listing 6.10 Example of Malware.Spyware Alert


```
{
  "Node": [
    {
      "SW": [
        "TippingPoint_NX_NGIPS"
      ],
      "Type": [
        "Datagram",
        "Content",
        "Protocol",
        "Signature",
        "Policy",
        "Heuristic"
      ],
      "Name": "cz.cesnet.ids_collector.tippingpoint"
    }
  ],
  "Category": [
    "Malware.Spyware"
  ],
  "Source": [
    {
      "Type": [
        "Botnet"
      ],
    }
```

```
    ],
  }
```

- Anomaly.Protocol - This category identifies anomalous network communication protocols that deviate from expected patterns. It may indicate potential security incidents, such as unauthorized protocol usage or attempts to bypass security controls.

Listing 6.11 Example of Anomaly.Protocol Alert

```
{
  "Node": [
    {
      "SW": [
        "TippingPoint_NX_NGIPS"
      ],
      "Type": [
        "Datagram",
        "Content",
        "Protocol",
        "Signature",
        "Policy",
        "Heuristic"
      ],
      "Name": "cz.cesnet.ids_collector.tippingpoint"
    }
  ],
  "Category": [
    "Anomaly.Protocol"
  ],
}
```

- Malware.Worm - This category represents a worm's presence or potential presence, a type of malicious software that self-replicates and spreads across networked systems without human intervention, aiming to compromise or disrupt the targeted systems.

Listing 6.12 Example of Malware.Worm Alert

```
{
  "Node": [
```

```
      {
        "SW": [
          "TippingPoint_NX_NGIPS"
        ],
        "Type": [
          "Datagram",
          "Content",
          "Protocol",
          "Signature",
          "Policy",
          "Heuristic"
        ],
        "Name": "cz.cesnet.ids_collector.tippingpoint"
      }
    ],
    "Category": [
      "Malware.Worm"
    ]
  }
```


- Other - This category refers to alerts that do not fit into predefined categories or require further analysis to determine their nature or impact.

Listing 6.13 Example of Other Alert


```
  {
    "Node": [
      {
        "SW": [
          "Suricata"
        ],
        "Type": [
          "Datagram",
          "Content",
          "Protocol",
          "Signature"
        ],
        "Name": "cz.cesnet.ids_collector.suricata"
```

```
      }
    ],
    "Category": [
      "Other"
    ]
  }
```


- Malware.Trojan - This category signifies the presence or potential presence of a Trojan horse, a type of malicious software disguised as legitimate software. Trojans typically perform unauthorized actions on a system, often creating a backdoor for remote attackers or stealing sensitive data.

Listing 6.14 Example of Malware.Trojan Alert


```
{
  "Node": [
    {
      "SW": [
        "TippingPoint_NX_NGIPS"
      ],
      "Type": [
        "Datagram",
        "Content",
        "Protocol",
        "Signature",
        "Policy",
        "Heuristic"
      ],
      "Name": "cz.cesnet.ids_collector.tippingpoint"
    }
  ],
  "Category": [
    "Malware.Trojan"
  ]
}
```


- Vulnerable.Config - This category indicates vulnerabilities or misconfigurations in a

system or network that attackers can exploit to gain unauthorized access or perform malicious actions.

Listing 6.15 Example of Vulnerable.Config Alert


```
{

  "Node": [
    {
      "SW": [
        "SSERV"
      ],
      "Type": [
        "External",
        "Recon"
      ],
      "Name": "cz.cesnet.ext.sserv"
    }
  ],
  "Category": [
    "Vulnerable.Config"
  ],
  "Source": [
    {
      "Note": "System name: 4",
      "Proto": [
        "udp",
        "ntp"
      ]
    }
  ]
}
```


- Abusive.Spam - This category relates to the presence or potential presence of spam emails or messaging that contains abusive or unsolicited content, often with malicious intent or attempts to deceive recipients.

Listing 6.16 Example of Abusive.Spam Alert

```
{
  "Node": [
    {
      "SW": [
        "Warden Filer"
      ],
      "Type": [
        "Relay"
      ],
      "Name": "cz.cesnet.supplier.warden_filer"
    }
  ],
  "Category": [
    "Abusive.Spam"
  ]
}
```


- Intrusion.Botnet - This category signifies a botnet's presence or potential presence, a network of compromised devices controlled by a malicious actor. Botnets are typically used for malicious activities, such as launching DDoS attacks, distributing malware, or harvesting sensitive information.

Listing 6.17 Example of Intrusion.Botnet Alert


```
{
  "Node": [
    {
      "SW": [
        "Nemea",
        "blacklistfilter"
      ],
      "Type": [
        "Flow",
        "Blacklist"
      ],
      "Name": "cz.casablanca.nemea.blacklistfilter"
    }
  ],
```

```
    "Category": [
      "Intrusion.Botnet"
    ],
    "Source": [
      {
        "InPacketsCount": 27,
        "OutPacketsCount": 53,
        "OutFlowCount": 1,
        "InByteCount": 1720,
        "OutByteCount": 73234,
        "Type": [
          "CC",
          "Botnet"
        ]
      }
    ],
    "ByteCount": 74954,
    "FlowCount": 2,
    "PacketCount": 80
  }
```


- Fraud.Phishing - This category denotes the presence or potential presence of phishing attempts involving malicious actors tricking individuals into revealing sensitive information, such as login credentials or financial details, by masquerading as trustworthy entities.

Listing 6.18 Example of Fraud.Phishing Alert


```
  {
    "Node": [
      {
        "SW": [
          "Warden Filer"
        ],
        "Type": [
          "Relay"
        ],
        "Name": "cz.cesnet.supplier.warden_filer"
```

```
      }
    ],
    "Category": [
      "Fraud.Phishing"
    ],
    "Source": [
      {
        "URL": [
          "URL3976"
        ],
        "Type": [
          "Phishing"
        ]
      }
    ]
  }
```


6.4 Data Visualization And Clustering

We initially explored data visualization to enhance our comprehension of the SABU dataset [71] and its inherent structure. Due to the dataset's high-dimensional characteristics, conventional plotting techniques would be inadequate. Hence, Principal Component Analysis (PCA), a widely used method for reducing dimensionality, was used to visually represent the data in a two-dimensional plane. A three-dimensional visualization was not necessary. After calculating the explained variance of each principal component (PC1: 35%, PC2: 30%, PC3: 14%) that is illustrated in table 6.2 and after visualizing it in a scree plot showcased in Figure 6.1, it was evident that the first two principal components (PC1 and PC2) accounted for a significant portion of the total variance in the data, summing to 65%. This justified the decision to use a two-dimensional visualization, as the third component (PC3) contributed only an additional 14% to the explained variance.

Table 6.2 Eigenvalues and Variance of the sampled data

Component	Eigenvalue	% of Variance	% of Variance Cumulative
PC1	7.613598	34.61%	34.61%
PC2	6.641740	30.19%	64.80%
PC3	3.065891	13.94%	78.73%
PC4	1.999905	9.09%	87.82%
PC5	0.999511	4.54%	92.37%

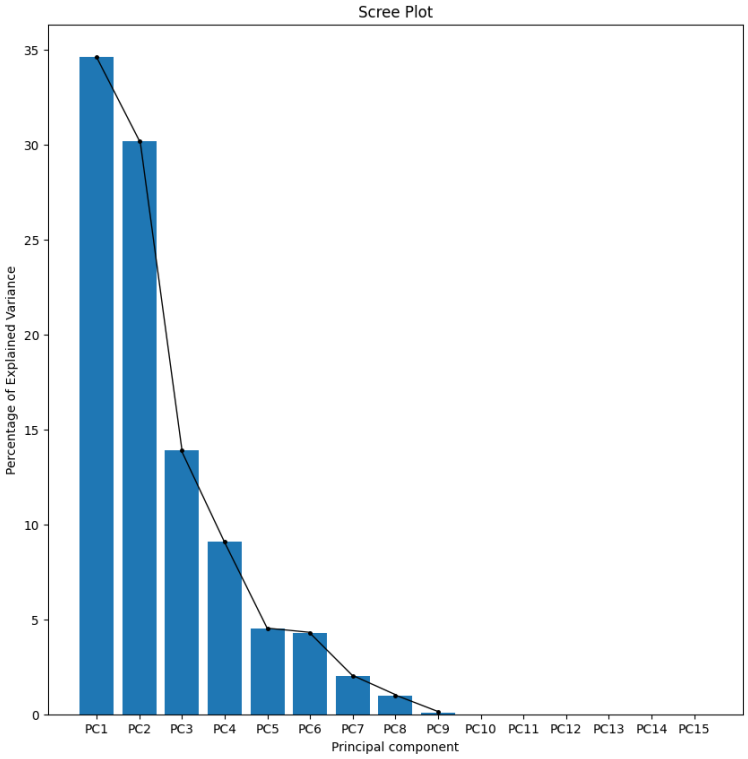

Figure 6.1 Scree Plot of the sampled data after PCA

In figure 6.3, each data point is colored depending on its category. The plot's absence of colors is explained by the concentration of different category data points in the same zone. Therefore, not all the colors shown in the legend are visible in the plot. The two dominant colors are the blue color that represents the Anomaly. Traffic attack category and the violet color that represents the Availability.DDoS attack category.

We can identify two to three clusters that could represent the similarity groups. The blue data points (Anomaly.Traffic) form the first cluster. The violet data points (Availability.DDoS) include the second cluster. The rest of the data points represent the third cluster. Another option is to merge the second and third clusters to have two clusters in total. These clusters

are potential similarity indicators in the argumentation process.

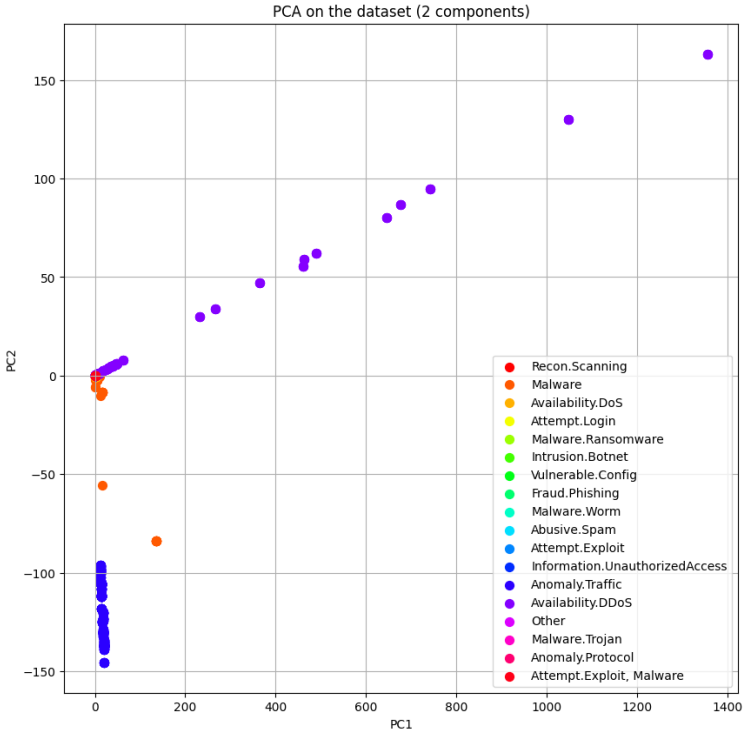

Figure 6.2 Scatter Plot of the sampled data after PCA

Table 6.3 illustrates the data features that each principal component is built on.

- PC1 : This column lists variables presumably contributing to the first principal component (PC1). The variables include 'FlowCount,' 'PacketCount,' 'Source_PortCount,' 'Source_ProtoCount,' 'Source_InterfaceCount,' 'Target_PortCount,' 'Target_ProtoCount,' 'Target_InterfaceCount,' 'Target_IP4Count', 'Source_Port,' and 'Target_Port.' These variables focus on basic network traffic metrics like flow, packet counts, and specifics regarding source and target ports, protocols, interfaces, and IP addresses.

- PC2: This column, in contrast, lists variables contributing to the second principal component (PC2). It includes 'ByteCount,' 'Source_InFlowCount,' 'Source_OutFlowCount,' 'Source_InByteCount,' 'Source_OutByteCount,' 'Source_InPacketCount,' 'Source_OutPacketCount,' 'Target_InPacketCount,' 'Target_InByteCount,' 'Target_InFlowCount,' 'Source_Proto,' and 'Target_Proto.' These variables emphasize the volume of data (in bytes) and more detailed aspects of the flow and packets, explicitly focusing on inbound and outbound traffic from both source and target perspectives.

Table 6.3 Variables contributing to PC1 and PC2

PC1	**PC2**
FlowCount	ByteCount
PacketCount	Source_InFlowCount
Source_PortCount	Source_OutFlowCount
Source_ProtoCount	Source_InByteCount
Source_InterfaceCount	Source_OutByteCount
Target_PortCount	Source_InPacketCount
Target_ProtoCount	Source_OutPacketCount
Target_InterfaceCount	Target_InPacketCount
Target_IP4Count	Target_InByteCount
Source_Port	Target_InFlowCount
Target_Port	Source_Proto
Target_Interface	Target_Proto

Due to the high density of data points in the initial scatter plot, specific categories were not adequately visualized, resulting in the absence of distinguishable colors. We adopted a characterization approach for each category using the same sampled data to address this issue. We computed the mean of each numerical field and determined the frequency of each categorical field value. This approach allowed us to obtain statistical summaries that provide insights into the central tendency and distribution of the numeric data and the prevalence of specific categorical values within each category. Principal Component Analysis (PCA) was applied to the characterized data, enabling a new scatter plot to be generated and examined. In Figure [Insert Figure Number], the scatter plot demonstrates that the categories "Availability.DDoS" and "Anomaly.Traffic" are noticeably separated from the remaining categories, which are predominantly concentrated within a single zone.

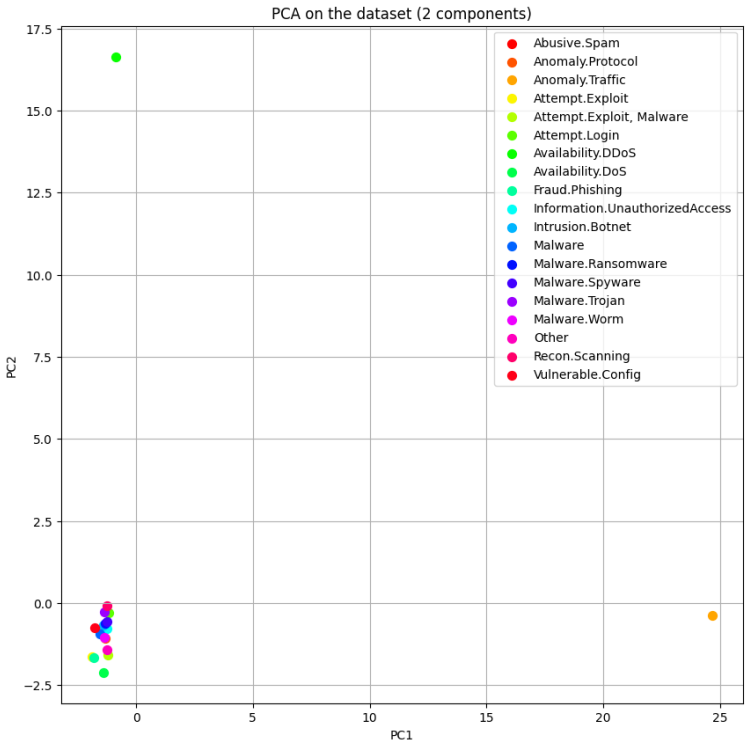

Figure 6.3 Scatter Plot of the sampled data characterization after PCA

We proceeded to employ clustering techniques to generate indicators of similarity. Creating a clustering model is crucial as it enables the generation of these indicators. The resulting similarity indicator is then utilized to determine the relationship (attack or support) between the initiating argument and the subsequently generated argument by X-Guard. This latter argument is intended to be transmitted to M-Guard, which operates on the blockchain platform, to construct the argumentation graph.

Two commonly used clustering techniques were used in this study: K-means [89] and DBSCAN (Density-Based Spatial Clustering of Applications with Noise) [90]. The research aimed to determine which model could provide the most precise similarity indicators while reducing conflicts related to the detection categories. To compare the performance of the K-means and DBSCAN models, we evaluated the similarity indicators generated by each model. We considered the quality of the indicators (how well they represented the similarity within the data) and the number of conflicts concerning the categories. The model that produced similarity indicators with high intra-indicator similarity and low inter-indicator similarity while minimizing category conflicts was considered the better model for our dataset. The K-means and DBSCAN models were comprehensively evaluated with the specified criteria. The outcomes of this comparison, in conjunction with our determination of the most appropriate

model for our dataset, are outlined in the subsequent table 6.4.

Table 6.4 Comparison of Clustering Models

Model	**Optimal Nbr of Clusters**	**Silhouette Score**	**Entropy**
KMeans [89]	3	0.90	0.376
DBSCAN [90]	2	0.91	0.003

The evaluation used three primary metrics: the optimal number of clusters, the silhouette score, and entropy.

The elbow method [91] was employed to determine the optimal number of clusters for the KMeans algorithm, while the DBSCAN algorithm was used for the same purpose in the density-based spatial clustering of applications with noise. The KMeans algorithm yielded an optimal number of clusters equal to 3, whereas the DBSCAN algorithm indicated that 2 clusters would be optimal.

The silhouette score, which quantifies the similarity of an object to its cluster relative to other clusters, exhibited a slightly higher value for DBSCAN (0.91) compared to KMeans (0.9). This suggests that DBSCAN demonstrated a somewhat superior performance in terms of clustering quality.

The entropy, which quantifies the impurity level or disorder within a given set of categories, exhibited a notably reduced value of 0.003 for the DBSCAN algorithm in contrast to the KMeans algorithm, which yielded a higher entropy of 0.376. This finding implies that DBSCAN exhibited greater efficacy in generating clusters characterized by a dominant category, a desirable outcome within our specific context, as it signifies a heightened degree of purity in the clustering procedure.

The DBSCAN clustering model was selected as the preferred choice based on the findings obtained. Despite possessing fewer clusters, it exhibited superior performance in both silhouette score and entropy, suggesting enhanced cluster quality and purity.

The categories are distributed in the DBSCAN-generated clusters, as shown in Table 6.5.

Table 6.5 Comparison of Clustering Models

Cluster number	Categories
1	Anomaly.Traffic
2	The rest of the SABU Dataset categories

6.5 Argumentation Result And Validation

We established an argumentation scenario with pre-determined expected results for testing and validating our implementation. This approach allows us to assess the performance and accuracy of our framework by comparing the actual outcomes against the anticipated outcomes.

6.5.1 Argumentation Scenario

To validate our implementation, we selected an alert from the SABU dataset not included in the sampled data for building the clustering model. Following the argument generation process outlined in Chapter 4 in section 4.2.1, we created an argument representing the initial argument that initiates the argumentation process.

Additionally, we introduced four other arguments into the scenario, each related to the same detection event as the initial argument. It is expected that these arguments should have the same similarity indicator as the initial argument. However, we purposely altered the similarity indicator in one of the arguments. Consequently, we anticipate that this altered argument will launch an attack against the initial argument and, in turn, be attacked by all other arguments.

By having prior knowledge of the expected results, we can validate the accuracy of our implementation by comparing the actual outcomes with the predicted results. This approach helps us assess the effectiveness and reliability of our argumentation framework.

The scenario is represented in the Table 6.6. A1 is the argument that initiated the argumentation process with "Anomaly.Traffic" as the detection Event and 1 as the similarity indicator. A2 is the argument that has the altered similarity indicator. The cross on the $\otimes$ column shows that A2 chooses to attack A1. On the other hand, A3, 4, and 5 decide to support ($\subseteq$) A1 as they have the same similarity indicator introduced by A1. From this scenario, we can manually build the argumentation graph using the definitions we introduced in section 4.2.1 of Chapter 4 and extract the preferred extension to validate with the automatically generated results as a next step.

Table 6.6 Argumentation example scenario

Argument ID	**Detection Event**	**Happens**	**Initiated**	$\otimes$	$\subseteq$	**SI**
A1	Anomaly. Traffic	Source:[Proto:['tcp'], IP4: ['69.255.87.252']], ByteCount:9713600, FlowCount:6071, PacketCount:242840	"DetectTime": "2019-03-17 T00:00:40+01:00"	-	-	1
A2	-	-	-	X	-	2
A3	-	-	-	-	X	1
A4	-	-	-	-	X	1
A5	-	-	-	-	X	1

First, we have $DetectionEvent(A1) \in \{Anomaly.Traffic\}$, $SI(A1) \neq SI(\{Anomaly.Traffic\})$ (as for the poisoned entity that is attacking the SI for that category is no longer 1 but 2) and $SI(A2) = SI(\{Anomaly.Traffic\})$. Therefore, based on Definition 4.2.2, $A2 \otimes A1$. As A2 attacks A1 ($A2 \otimes A1$) and A3 supports A1 ($A3 \subseteq A1$), A3 attacks A2 ($A3 \otimes A2$) by applying Defintion 4.2.2. Applying the same rule, A4 and A5 also attack A2 ($A4 \subseteq A1$, $A5 \subseteq A1$).

We can then introduce our IDSAF by following the definition 4.2.2. $N = \{A1, A2, A3, A4, A5\}$, $R = \{(A2, A1), (A3, A2), (A4, A2), (A5, A2)\}$.

We can also identify the preferred extension as defined by Dung. {A1, A3, A4, A5} is the one in this scenario.

This theoretical result will be compared to the automatically generated result to validate our implementation.

6.5.2 Scenario simulation And Validation

Using the DBSCAN clustering model, we launched our argumentation process in an environment composed of five X-Guards, and we were able to visualize the attack graph and extract the preferred extension of the argumentation Framework on our M-Guard.

In the simulated scenario (Table 6.6), an argumentation process was initiated among five X-Guard nodes, wherein one of the nodes possessed a clustering model that was intentionally poisoned. The argumentation process was started as a result of argument A1, which was generated based on the occurrence of a detection event. The argument denoted as A1 was subsequently distributed to the remaining nodes for assessment. Each node ran its similarity indicator after receiving input A1. The poisoned node attacked argument A1 because its

faulty clustering model caused it to provide a deviant similarity indication. In graph 6.4, this assault is shown by the argument labeled "A2".

Concurrently, the remaining nodes employed accurate clustering models to assess the validity of the attack argument A2 based on their respective similarity indicators. The nodes initiated counter-attacks against argument A2 due to observing its inconsistency with their indicators. The counter-attacks are denoted as arguments A3, A4, and A5 in the graphical representation in figure 6.4.

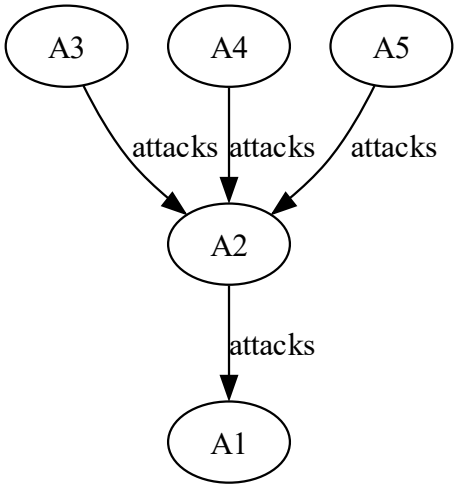

Figure 6.4 Simulation Attack Graph

By comparing the graph 6.4 automatically generated and the one we theoretically identified: $N = \{A1, A2, A3, A4, A5\}$, $R = \{(A2, A1), (A3, A2), (A4, A2), (A5, A2)\}$, we can confirm that both results are the same and the automatically generated results are valid.

The preferred extension was also computed automatically as the collection of arguments A1, A3, A4, A5. This set represents the collectively deemed acceptable arguments based on the argumentation process among the X-Guard nodes. Argument A1, the initial argument in the argumentation process, remains unchallenged by any other argument in the preferred extension. Arguments A3, A4, and A5 attack A2, which attacks A1. It is worth noting that the preferred extension exhibits an absence of internal conflicts among its arguments, thereby explaining the actual extension result.

Changing the initial argument consistently leads to the same flow of events in different

scenarios, including a sub-scenario where a poisonous node initiates the argumentation and results in a collective attack on the poisoned argument; it provides strong evidence for the validity and scalability of our argumentation framework. This suggests that the framework's principles and mechanisms are robust and adaptable across different scenarios within the SABU dataset.

6.6 Conclusion

As presented in the chapter, the XM-Guards framework comprehensively incorporates tools, techniques, and approaches to address cybersecurity threats. Leveraging a private Ethereum blockchain, Docker, Web3, and Python, the framework ensures the necessary infrastructure and programming environments are available for development and testing.

The framework utilizes the SABU dataset to understand and analyze network traffic data. Data visualization and clustering techniques were applied to the dataset to generate a similarity indicator. This indicator played a crucial role in implementing the argumentation framework.

The implementation of the argumentation framework was then tested on a scenario extracted from the SABU dataset. The theoretical and automated results were compared to validate the framework's validity and effectiveness. The generated graph and its preferred extension showcased the framework's ability to capture the argumentation process accurately.

CHAPTER 7 CONCLUSION

intrusion detection systems are digital infrastructures that protect against malicious activities. With the continuous evolution of cyber threats into more intricate forms, there is an urgent requirement to develop and improve IDS functionalities. Our work presents an innovative methodology that integrates the resilient properties of blockchain technology with the fluid capabilities of argumentation-based reasoning. By combining these elements, a novel approach is taken to meet the challenges intrinsic to conventional intrusion detection systems and combat emergent cyber threats. The following sections constitute an exhaustive overview of the study, including a summary of the work, its limitations, and potential directions for future research.

7.1 Summary of Works

Our thesis presents a comprehensive study on enhancing decision-making in Collaborative Intrusion Detection Systems (CIDS) through a formal argumentation framework. It validates the XM-Guards system using real-world data. The core of our research lies in integrating a structured argumentation framework in CIDS, significantly improving decision-making processes. This framework facilitates capturing and analyzing the entire spectrum of alerts and arguments exchanged between different detection nodes. By constructing a detailed graph from these interactions, we successfully isolate arguments free from conflicts, offering a more precise and dependable view of potential intrusions. This methodology enables experts to extract non-conflicting arguments from a complex array of data, leading to more informed decision-making during intrusion events. The strength of this approach is its structured nature, allowing for systematic evaluation and integration of multi-source data, which yields a more accurate and comprehensive understanding of security threats.

To further substantiate our research, we implemented and tested the XM-Guards system using data from the SABU Dataset. This real-world application confirmed the practicality and effectiveness of XM-Guards in real-life scenarios, demonstrating its capability to process and analyze data in conditions that mirror actual operations. The employment of real-world data is crucial as it provides concrete evidence of XM-Guards' operational capabilities and performance under realistic conditions. This reinforces the theoretical underpinnings of the XM-Guards system and underscores its readiness for practical deployment in real-world intrusion detection scenarios, making a significant contribution to cybersecurity.

7.2 Limitations

Despite introducing an innovative methodology, the research had certain limitations. The main objective did not involve showcasing the potential of blockchain technology for improving the sharing and distribution of detection events. The effectiveness of the argumentation process heavily relied on the quality of the clustering models that produced similarity indicators. The research primarily concentrated on the similarity metric and overlooked alternative indicators such as temporal, spatial, and attack patterns. Additionally, the study did not explore other real-world datasets apart from the SABU dataset.

7.3 Future Research

The investigation conducted in this study has yielded significant findings regarding incorporating blockchain technology into argumentation-based reasoning. Still, it has also shed light on a promising avenue for additional scholarly inquiry and novel advancements. The integration of argumentation-based reasoning and blockchain technology has exhibited considerable potential in augmenting intrusion detection systems.

Blockchain technology's decentralized and transparent characteristics present a methodology for distributing detection events. Although the present study has examined its potential, there are multiple prospects to explore and validate further the profound ways blockchain technology can revolutionize the dissemination and validation of detection events.

Additionally, the argumentation process can be significantly enhanced by integrating supplementary metrics. The system's dependability and predictive capabilities can be substantially improved by incorporating metrics such as TSI, SSI, and the potentially effective Attack Patterns Indicator (API). Nevertheless, an ongoing obstacle is the absence of a dataset containing notifications for multi-stage attacks, which is critical for the efficient development and verification of the API. It is essential to address this gap to fully utilize the capabilities of the API in understanding multi-stage attacks for future research.

Moreover, the field of clustering models offers a wide range of options, each with unique advantages and qualities. While this investigation demonstrated the effectiveness of the DBSCAN clustering model, conducting experiments with various clustering models can provide more diverse results and potentially valuable insights. By exploring clustering models other than K-means and DBSCAN, we can improve the generation of similarity indicators and achieve a more comprehensive understanding of intrusion patterns.

Finally, to ensure the validity and applicability of the methodology, it is intended to test it

on different datasets, including various intrusion scenarios, network topologies, and architectures. Doing so can not only validate the effectiveness of the methodology but also provide insights into potential areas of improvement and refinement.

Fundamentally, this study has established a groundwork, preparing for more advanced intrusion detection investigations. The gathered insights and identified opportunities suggest a future where intrusion detection systems are more effective, proactive, resilient, and predictable.

REFERENCES

[1] J. Kašpar, "Experimenting with the aida framework."

[2] W. Li, W. Meng, and L. F. Kwok, "Surveying trust-based collaborative intrusion detection: State-of-the-art, challenges and future directions," *IEEE Communications Surveys Tutorials*, vol. 24, no. 1, pp. 280–305, 2022.

[3] E. Vasilomanolakis, S. Karuppayah, M. Mühlhäuser, and M. Fischer, "Taxonomy and survey of collaborative intrusion detection," *ACM Computing Surveys (CSUR)*, vol. 47, no. 4, pp. 1–33, 2015.

[4] Z. Zheng, S. Xie, H.-N. Dai, W. Chen, X. Chen, J. Weng, and M. Imran, "An overview on smart contracts: Challenges, advances and platforms," *Future Generation Computer Systems*, vol. 105, pp. 475–491, 2020.

[5] K. Wüst and A. Gervais, "Do you need a blockchain?" in *2018 Crypto Valley Conference on Blockchain Technology (CVCBT)*, 2018, pp. 45–54.

[6] "Alarming cyber statistics for mid-year 2022 that you need to know," Forbes, Jun 3, 2022. [Online]. Available: https://www.forbes.com/sites/chuckbrooks/2022/06/03/alarming-cyber-statistics-for-mid-year-2022-that-you-need-to-know/?sh=68c733497864

[7] "Check point research: Third quarter of 2022 reveals increase in cyberattacks and unexpected developments in global trends," CheckPoint, 2022. [Online]. Available: https://blog.checkpoint.com/2022/10/26/third-quarter-of-2022-reveals-increase-in-cyberattacks/

[8] "How aligning security and the business creates cyber resilience," Accenture, 2021. [Online]. Available: https://www.accenture.com/_acnmedia/PDF-165/Accenture-State-Of-Cybersecurity-2021.pdf#zoom=40

[9] "Ransomware recovery cost reaches nearly $2 million, more than doubling in a year, sophos survey shows," sophos, April 27, 2021. [Online]. Available: https://www.sophos.com/en-us/press-office/press-releases/2021/04/ransomware-recovery-cost-reaches-nearly-dollar-2-million-more-than-doubling-in-a-year

[10] P. O'Kane, S. Sezer, and D. Carlin, "Evolution of ransomware," *Iet Networks*, vol. 7, no. 5, pp. 321–327, 2018.

[11] A. Khraisat, I. Gondal, P. Vamplew, and J. Kamruzzaman, "Survey of intrusion detection systems: techniques, datasets and challenges," *Cybersecurity*, vol. 2, no. 1, pp. 1–22, 2019.

[12] M. Gil Pérez, F. Gomez Marmol, G. Martinez Perez, and A. F. Skarmeta Gómez, "Repcidn: A reputation-based collaborative intrusion detection network to lessen the impact of malicious alarms," *Journal of Network and Systems Management*, vol. 21, pp. 128–167, 2013.

[13] N. Alexopoulos, E. Vasilomanolakis, N. R. Ivánkó, and M. Mühlhäuser, "Towards blockchain-based collaborative intrusion detection systems," in *Critical Information Infrastructures Security: 12th International Conference, CRITIS 2017, Lucca, Italy, October 8-13, 2017, Revised Selected Papers 12*. Springer, 2018, pp. 107–118.

[14] M. Özalp, C. Karakuzu, and A. Zengin, "Distributed intrusion detection systems: A survey," *Academic Perspective Procedia*, vol. 2, no. 3, pp. 400–407, 2019.

[15] W. Meng, W. Li, Y. Xiang, and K.-K. R. Choo, "A bayesian inference-based detection mechanism to defend medical smartphone networks against insider attacks," *Journal of Network and Computer Applications*, vol. 78, pp. 162–169, 2017.

[16] W. Li, Y. Meng, and L.-F. Kwok, "Enhancing trust evaluation using intrusion sensitivity in collaborative intrusion detection networks: feasibility and challenges," in *2013 ninth international conference on computational intelligence and security*. IEEE, 2013, pp. 518–522.

[17] C. J. Fung and Q. Zhu, "Facid: A trust-based collaborative decision framework for intrusion detection networks," *Ad Hoc Networks*, vol. 53, pp. 17–31, 2016.

[18] W. Li, W. Meng, J. Parra-Arnau, and K.-K. R. Choo, "Enhancing challenge-based collaborative intrusion detection against insider attacks using spatial correlation," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2021, pp. 1–8.

[19] E. Ferretti, L. H. Tamargo, A. J. García, M. L. Errecalde, and G. R. Simari, "An approach to decision making based on dynamic argumentation systems," *Artificial intelligence*, vol. 242, pp. 107–131, 2017.

[20] P. Baroni, G. Boella, F. Cerutti, M. Giacomin, L. Van Der Torre, and S. Villata, "On the input/output behavior of argumentation frameworks," *Artificial Intelligence*, vol. 217, pp. 144–197, 2014.

[21] M. Baykara and R. Daş, "A survey on potential applications of honeypot technology in intrusion detection systems," *International Journal of Computer Networks and Applications (IJCNA)*, vol. 2, no. 5, pp. 203–211, 2015.

[22] G. Folino and P. Sabatino, "Ensemble based collaborative and distributed intrusion detection systems: A survey," *Journal of Network and Computer Applications*, vol. 66, pp. 1–16, 2016.

[23] R. Anderson, *Security engineering: a guide to building dependable distributed systems*. John Wiley & Sons, 2020.

[24] E. Vasilomanolakis, M. Stahn, C. G. Cordero, and M. Mühlhäuser, "Probe-response attacks on collaborative intrusion detection systems: Effectiveness and countermeasures," in *2015 IEEE Conference on Communications and Network Security (CNS)*. IEEE, 2015, pp. 699–700.

[25] C. Birkinshaw, E. Rouka, and V. G. Vassilakis, "Implementing an intrusion detection and prevention system using software-defined networking: Defending against port-scanning and denial-of-service attacks," *Journal of Network and Computer Applications*, vol. 136, pp. 71–85, 2019.

[26] A. Iliev, N. Kyurkchiev, A. Rahnev, and T. Terzieva, *Some models in the theory of computer viruses propagation*. LAP LAMBERT Academic Publishing Saarbrucken, Germany, 2019.

[27] M. Baezner and P. Robin, "Stuxnet," ETH Zurich, Tech. Rep., 2017.

[28] R. Vishwakarma and A. K. Jain, "A survey of ddos attacking techniques and defence mechanisms in the iot network," *Telecommunication systems*, vol. 73, no. 1, pp. 3–25, 2020.

[29] B. D. Deebak and F. Al-Turjman, "A hybrid secure routing and monitoring mechanism in iot-based wireless sensor networks," *Ad Hoc Networks*, vol. 97, p. 102022, 2020.

[30] S. T. Patel and N. H. Mistry, "A review: Sybil attack detection techniques in wsn," in *2017 4th International conference on electronics and communication systems (ICECS)*. IEEE, 2017, pp. 184–188.

[31] S. Trifunovic and A. Hossmann-Picu, "Stalk and lie—the cost of sybil attacks in opportunistic networks," *Computer Communications*, vol. 73, pp. 66–79, 2016.

[32] W. Meng, W. Li, L. T. Yang, and P. Li, "Enhancing challenge-based collaborative intrusion detection networks against insider attacks using blockchain," *International Journal of Information Security*, vol. 19, pp. 279–290, 2020.

[33] W. Li, W. Meng, L.-F. Kwok, and H. Horace, "Enhancing collaborative intrusion detection networks against insider attacks using supervised intrusion sensitivity-based trust management model," *Journal of Network and Computer Applications*, vol. 77, pp. 135–145, 2017.

[34] W. Fang, W. Zhang, W. Chen, T. Pan, Y. Ni, and Y. Yang, "Trust-based attack and defense in wireless sensor networks: a survey," *Wireless Communications and Mobile Computing*, vol. 2020, pp. 1–20, 2020.

[35] W. Meng, X. Luo, W. Li, and Y. Li, "Design and evaluation of advanced collusion attacks on collaborative intrusion detection networks in practice," in *2016 IEEE Trustcom/BigDataSE/ISPA*. IEEE, 2016, pp. 1061–1068.

[36] C. J. Fung, D. Y. Lam, and R. Boutaba, "Revmatch: An efficient and robust decision model for collaborative malware detection," in *2014 IEEE Network Operations and Management Symposium (NOMS)*. IEEE, 2014, pp. 1–9.

[37] N. Kumar and N. Chilamkurti, "Collaborative trust aware intelligent intrusion detection in vanets," *Computers & Electrical Engineering*, vol. 40, no. 6, pp. 1981–1996, 2014.

[38] W. Meng, W. Li, Y. Wang, and M. H. Au, "Detecting malicious nodes in medical smartphone networks through euclidean distance-based behavioral profiling," in *Cyberspace Safety and Security: 9th International Symposium, CSS 2017, Xi'an China, October 23–25, 2017, Proceedings*. Springer, 2017, pp. 163–175.

[39] G. Rajeshkumar and K. Valluvan, "An energy aware trust based intrusion detection system with adaptive acknowledgement for wireless sensor network," *Wireless Personal Communications*, vol. 94, pp. 1993–2007, 2017.

[40] W. Meng, W. Li, C. Su, J. Zhou, and R. Lu, "Enhancing trust management for wireless intrusion detection via traffic sampling in the era of big data," *Ieee Access*, vol. 6, pp. 7234–7243, 2017.

[41] W. Meng, "Intrusion detection in the era of iot: Building trust via traffic filtering and sampling," *Computer*, vol. 51, no. 7, pp. 36–43, 2018.

[42] W. Meng, W. Li, and J. Zhou, "Enhancing the security of blockchain-based software defined networking through trust-based traffic fusion and filtration," *Information Fusion*, vol. 70, pp. 60–71, 2021.

[43] W. Li, W. Meng, J. Parra-Arnau, and K.-K. R. Choo, "Enhancing challenge-based collaborative intrusion detection against insider attacks using spatial correlation," in *2021 IEEE Conference on Dependable and Secure Computing (DSC)*. IEEE, 2021, pp. 1–8.

[44] W. Li, W. Meng, and L. T. Yang, "Enhancing trust-based medical smartphone networks via blockchain-based traffic sampling," in *2021 IEEE 20th International Conference on Trust, Security and Privacy in Computing and Communications (TrustCom)*. IEEE, 2021, pp. 122–129.

[45] S. Nakamoto, "Bitcoin: a peer-to-peer electronic cash system, oct. 2008," *URL http://www. bitcoin. org/bitcoin. pdf.(cited on pp. 15 and 87)*, 2017.

[46] D. Puthal, N. Malik, S. P. Mohanty, E. Kougianos, and G. Das, "Everything you wanted to know about the blockchain: Its promise, components, processes, and problems," *IEEE Consumer Electronics Magazine*, vol. 7, no. 4, pp. 6–14, 2018.

[47] V. Buterin *et al.*, "A next-generation smart contract and decentralized application platform," *white paper*, vol. 3, no. 37, pp. 2–1, 2014.

[48] P. Fairley, "Ethereum will cut back its absurd energy use," *IEEE spectrum*, vol. 56, no. 1, pp. 29–32, 2018.

[49] J. H. Battisti, G. P. Koslovski, M. A. Pillon, C. C. Miers, and N. M. Gonzalez, "Analysis of an ethereum private blockchain network hosted by virtual machines against internal dos attacks," in *International Conference on Advanced Information Networking and Applications*. Springer, 2022, pp. 479–490.

[50] K. Toyoda, K. Machi, Y. Ohtake, and A. N. Zhang, "Function-level bottleneck analysis of private proof-of-authority ethereum blockchain," *IEEE Access*, vol. 8, pp. 141 611–141 621, 2020.

[51] G. Zyskind, O. Nathan, and A. Pentland, "Enigma: Decentralized computation platform with guaranteed privacy," *arXiv preprint arXiv:1506.03471*, 2015.

[52] D. Shrier, W. Wu, and A. Pentland, "Blockchain & infrastructure (identity, data security)," *Massachusetts Institute of Technology-Connection Science*, vol. 1, no. 3, pp. 1–19, 2016.

[53] M. Mihaylov, S. Jurado, K. Van Moffaert, N. Avellana, and A. Nowé, "Nrg-x-change-a novel mechanism for trading of renewable energy in smart grids," in *International Conference on Smart Grids and Green IT Systems*, vol. 2. SciTePress, 2014, pp. 101–106.

[54] M. Mihaylov, S. Jurado, N. Avellana, K. Van Moffaert, I. M. de Abril, and A. Nowé, "Nrgcoin: Virtual currency for trading of renewable energy in smart grids," in *11th International conference on the European energy market (EEM14)*. IEEE, 2014, pp. 1–6.

[55] W. Li, Y. Wang, J. Li, and M. H. Au, "Towards blockchained challenge-based collaborative intrusion detection," in *Applied Cryptography and Network Security Workshops: ACNS 2019 Satellite Workshops, SiMLA, Cloud S&P, AIBlock, and AIoTS, Bogota, Colombia, June 5–7, 2019, Proceedings 17*. Springer, 2019, pp. 122–139.

[56] W. Li, J. Tan, and Y. Wang, "A framework of blockchain-based collaborative intrusion detection in software defined networking," in *Network and System Security: 14th International Conference, NSS 2020, Melbourne, VIC, Australia, November 25–27, 2020, Proceedings 14*. Springer, 2020, pp. 261–276.

[57] E. Vasilomanolakis, S. M. Habib, P. Milaszewicz, R. S. Malik, and M. Mühlhäuser, "Towards trust-aware collaborative intrusion detection: challenges and solutions," in *Trust Management XI: 11th IFIP WG 11.11 International Conference, IFIPTM 2017, Gothenburg, Sweden, June 12-16, 2017, Proceedings 11*. Springer, 2017, pp. 94–109.

[58] P. M. Dung, "On the acceptability of arguments and its fundamental role in nonmonotonic reasoning, logic programming and n-person games," *Artificial intelligence*, vol. 77, no. 2, pp. 321–357, 1995.

[59] K. Čyras, K. Satoh, and F. Toni, "Explanation for case-based reasoning via abstract argumentation," in *Computational Models of Argument*. IOS Press, 2016, pp. 243–254.

[60] J. Rowe, K. Levitt, S. Parsons, E. Sklar, A. Applebaum, and S. Jalal, "Argumentation logic to assist in security administration," in *Proceedings of the 2012 New Security Paradigms Workshop*, 2012, pp. 43–52.

[61] T. Bouyahia, F. Autrel, N. Cuppens-Boulahia, and F. Cuppens, "Context aware intrusion response based on argumentation logic," in *Risks and Security of Internet and Systems: 10th International Conference, CRiSIS 2015, Mytilene, Lesbos Island, Greece, July 20-22, 2015, Revised Selected Papers 10*. Springer, 2016, pp. 91–106.

[62] A. K. Bandara, A. Kakas, E. C. Lupu, and A. Russo, "Using argumentation logic for firewall policy specification and analysis," in *Large Scale Management of Distributed Systems: 17th IFIP/IEEE International Workshop on Distributed Systems: Operations and Management, DSOM 2006, Dublin, Ireland, October 23-25, 2006. Proceedings 17*. Springer, 2006, pp. 185–196.

[63] E. Karafili, K. Spanaki, and E. C. Lupu, "An argumentation reasoning approach for data processing," *Computers in Industry*, vol. 94, pp. 52–61, 2018.

[64] L. Yu, M. Zichichi, R. Markovich, and A. Najjar, "Intelligent human-input-based blockchain oracle (ihibo)." in *ICAART (1)*, 2022, pp. 515–526.

[65] W. Meng, W. Li, and L. Zhu, "Enhancing medical smartphone networks via blockchain-based trust management against insider attacks," *IEEE Transactions on Engineering Management*, vol. 67, no. 4, pp. 1377–1386, 2019.

[66] R. Aydoğan, N. Criado, J. Lang, V. Sanchez-Anguix, and M. Serramia, *PRIMA 2022: Principles and Practice of Multi-Agent Systems: 24th International Conference, Valencia, Spain, November 16–18, 2022, Proceedings*. Springer Nature, 2022, vol. 13753.

[67] M. Pataky and D. P. Gruska, "Multi-agent heterogeneous intrusion detection system." in *CS&P*, 2014, pp. 184–195.

[68] I. A. Saeed, A. Selamat, M. F. Rohani, O. Krejcar, and J. A. Chaudhry, "A systematic state-of-the-art analysis of multi-agent intrusion detection," *IEEE Access*, vol. 8, pp. 180 184–180 209, 2020.

[69] S. Doutre, M. Lafages, and M.-C. Lagasquie-Schiex, "A distributed and clustering-based algorithm for the enumeration problem in abstract argumentation," in *PRIMA 2019: Principles and Practice of Multi-Agent Systems: 22nd International Conference, Turin, Italy, October 28–31, 2019, Proceedings 22*. Springer, 2019, pp. 87–105.

[70] R. Miller and M. Shanahan, *The event calculus in classical logic-alternative axiomatisations*. Linköping University Electronic Press, 1999.

[71] M. Husák, M. Žádník, V. Bartoš, and P. Sokol, "Dataset of intrusion detection alerts from a sharing platform," *Data in Brief*, vol. 33, p. 106530, 2020.

[72] H. T. Elshoush and I. M. Osman, "Alert correlation in collaborative intelligent intrusion detection systems—a survey," *Applied Soft Computing*, vol. 11, no. 7, pp. 4349–4365, 2011.

[73] A. Aleroud and G. Karabatis, "Contextual information fusion for intrusion detection: a survey and taxonomy," *Knowledge and Information Systems*, vol. 52, pp. 563–619, 2017.

[74] M. Husák and J. Kašpar, "Aida framework: real-time correlation and prediction of intrusion detection alerts," in *Proceedings of the 14th international conference on availability, reliability and security*, 2019, pp. 1–8.

[75] [Online]. Available: https://idea.cesnet.cz/en/index

[76] [Online]. Available: https://kafka.apache.org/

[77] May 2022. [Online]. Available: https://www.docker.com/

[78] S. Voshmgir, *Token Economy: How the Web3 reinvents the internet.* Token Kitchen, 2020, vol. 2.

[79] Aug 2023. [Online]. Available: https://www.python.org/

[80] [Online]. Available: https://web3py.readthedocs.io/en/latest/providers.html

[81] [Online]. Available: https://graphviz.readthedocs.io/en/stable/

[82] T. Cejka, V. Bartos, M. Svepes, Z. Rosa, and H. Kubatova, "Nemea: a framework for network traffic analysis," in *2016 12th International Conference on Network and Service Management (CNSM).* IEEE, 2016, pp. 195–201.

[83] Sep 2023. [Online]. Available: https://suricata.io/

[84] [Online]. Available: https://archiv.cesnet.cz/doc/techzpravy/2004/ftas-arch/

[85] J. Franco, A. Aris, B. Canberk, and A. S. Uluagac, "A survey of honeypots and honeynets for internet of things, industrial internet of things, and cyber-physical systems," *IEEE Communications Surveys & Tutorials*, vol. 23, no. 4, pp. 2351–2383, 2021.

[86] [Online]. Available: https://labrea.sourceforge.io/labrea-info.html

[87] [Online]. Available: https://www.flowmon.com/en/products/software-modules/anomaly-detection-system

[88] M. Husák, M. Drašar *et al.*, "Flow-based monitoring of honeypots," in *7th International Conference on Security and Protection of Information (SPI 2013)*, 2013.

[89] J. A. Hartigan and M. A. Wong, "Algorithm as 136: A k-means clustering algorithm," *Journal of the royal statistical society. series c (applied statistics)*, vol. 28, no. 1, pp. 100–108, 1979.

[90] D. Birant and A. Kut, "St-dbscan: An algorithm for clustering spatial–temporal data," *Data & knowledge engineering*, vol. 60, no. 1, pp. 208–221, 2007.

[91] M. Syakur, B. Khotimah, E. Rochman, and B. D. Satoto, "Integration k-means clustering method and elbow method for identification of the best customer profile cluster," in *IOP conference series: materials science and engineering*, vol. 336. IOP Publishing, 2018, p. 012017.